G. Tatzl

Der Taschenrechner als Mini-Computer

1

Optimale Nutzung programmierbarer Elektronenrechner

Tatzl · Taschenrechner 1

Dipl.-Ing. Gerfried Tatzl

Der Taschenrechner als Mini-Computer

Band 1

Optimale Nutzung programmierbarer Elektronenrechner

Band 2

Kalkulation in Produktionsbetrieben mit programmierbaren Elektronenrechnern

Band 3

Auswertung von Baustoffprüfungen mit programmierbaren Elektronenrechnern

Band 4

Anlagenbuchhaltung und kalkulatorische Kostenermittlung mit programmierbaren Elektronenrechnern

BAUVERLAG GMBH · WIESBADEN UND BERLIN

Dipl.-Ing. Gerfried Tatzl

Der Taschenrechner als Mini-Computer

Band 1

Optimale Nutzung programmierbarer Elektronenrechner

BAUVERLAG GMBH · WIESBADEN UND BERLIN

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Tatzl, Gerfried:
Der Taschenrechner als Mini-Computer / Gerfried Tatzl.
– Wiesbaden, Berlin · Bauverlag.

Bd. 1. Optimale Nutzung programmierbarer
Elektronenrechner – 1979

ISBN 3-7625-1213-3

Das Werk ist urheberrechtlich geschützt.
Die dadurch begründeten Rechte, insbesondere die der Übersetzung
des Nachdruckes, der Entnahme von Abbildungen, der Funksendung, der Wiedergabe
auf photomechanischem (Fotokopie, Mikrokopie) oder ähnlichem Wege
und der Speicherung in Datenverarbeitungsanlagen,
bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten.

© 1979 Bauverlag GmbH · Wiesbaden und Berlin

Gesamtherstellung: Druckhaus Hans Meister KG, Kassel

ISBN 3–7625–1213–3

Vorwort

Die Rasanz der technischen Entwicklung ist derart atemberaubend, sodaß man sich fragen muß, wie der Mensch damit fertig wird. Wird er sich auf die neuen Gegebenheiten einstellen können? Darauf wird der Mensch eine Antwort kaum zu geben in der Lage seln. Es ist aber seiner Mitentscheidung anvertraut, sich an der Entwicklung zu orientieren und diese langsam zu verkraften.

In der Natur ist es schon so eingerichtet, daß nur der Stärkste die größten Überlebenschancen eingeräumt erhält. In der Rechentechnik scheint es mir nicht anders zu sein. Seit Osterloh im Herbst 1976 die Broschüre „Die Wahl des richtigen Taschenrechners" veröffentlicht hat, liegen Angaben der prominentesten Hersteller programmierbarer Taschenrechner vor, daß bis Ende des Jahres 1977 in ihren Firmen ein Umsatz von programmierbaren Modellen zu erwarten war, der genau so groß war, wie jener der übrigen Rechnertypen elektronischer Bauart. Sehr weit scheinen wir damit dem Zeitpunkt nicht mehr entfernt zu sein, zu dem der Spruch „Jedermann seinen eigenen Computer" voll in Erfüllung geht.

Osterloh hat also 1976 mit dem TAR den Taschenrechner schlechthin vorgestellt. Diese Species hat sich inzwischen wie eine Epidemie ausgebreitet und so rasend vermehrt, wie man es im allgemeinen von Hasen erwartet. Und das nicht nur mengen-, sondern auch qualitätsmäßig.

Daher war es nicht wunderlich und nachgerade auch zu erwarten, daß dem TAR sehr bald ein PROTAR folgen würde, der Taschenrechner hatte binnen kurzem einen Bruder bekommen, den programmierbaren Taschenrechner, den PROTAR. Dieser anspruchsvollere jüngere Bruder braucht nun aber wesentlich mehr Betreuung, um voll in der Gesellschaft der Rechengeräte anerkannt zu werden. Ohne Übertreibung könnte man sagen, daß sich dieser neue Sproß an die unterste Stelle der elektronischen Datenverarbeitung überhaupt gestellt hat und damit eine ausgezeichnete Ergänzung für alles bietet, für das sich eine größere Rechenanlage nicht rentiert. Meines Erachtens ist PROTAR wie geschaffen, auch für die kleinen und kleinsten Probleme, die eine gewisse Häufigkeit aufweisen, eine Lösungsmöglichkeit darzustellen. In diesem Sinne sei mit PROTAR jenen ein Gerät in die Hand gegeben, mit dem sie sich an Dinge heranwagen können, für die es bisher im allgemeinen nur die Lösung „zu Fuß" gegeben hat.

Nebenbei bemerkt, wäre es wünschenswert, bereits vor dem Kauf eines PROTARs über die verschiedenen Möglichkeiten informiert zu werden, das heißt, ein potentieller Käufer sollte vor dem Kauf eines Rechners über einzelne Modelle und über die Rechensysteme selbst aufgeklärt werden. Insofern wird diesem Buch eine entsprechende Verbreitung gewünscht.

Juli 1979 Gerfried Tatzl

Inhaltsverzeichnis

1. Einleitung 9

2. Der Rechner 14
   2.1. Aufbau eines Rechners 15
   2.2. Rechenlogik 17
      2.2.1. Die algebraische Logik 17
      2.2.2. Die umgekehrte polnische Notation 19
      2.2.3. Unterschiede 22
      2.2.4. Das Zählen beginnt bei 0 22
   2.3. Tastenfeld 23
      2.3.1. Zahlenfeld 26
      2.3.2. Mathematische Funktionen 26
      2.3.3. Prozentrechnung 29
      2.3.4. Speicherbefehle 29
      2.3.5. Programmbefehle 31
      2.3.6. Sonstige Tasten 33
   2.4. Ausgabe 34
      2.4.1. Anzeige 35
      2.4.2. Drucker 35
   2.5. Speicher 36
      2.5.1. Programmspeicher 36
      2.5.2. Datenspeicher 37
      2.5.3. Erweiterungen des Datenspeicherbereiches 38
      2.5.4. Sonstige Speicher 43
      2.5.5. Mehrfachverwendung von Speichern 44
   2.6. Besonderheiten programmierbarer Rechner 44
      2.6.1. DSZ – der automatische Zähler 44
      2.6.2. Indiziertes Rechnen 44
      2.6.3. Flags 46
      2.6.4. Konstantspeicher 46
      2.6.5. Last X-Register 47
      2.6.6. Magnetstreifen 47
      2.6.7. Listungen 47
      2.6.8. Module 48
      2.6.9. In letzter Minute 48
      2.6.10. Rechen- und Arbeitsgeschwindigkeit 49

3. Elemente der Programmierung 51
   3.1. Symbole 52
   3.2. Normaler Ablauf 54
   3.3. Programm–Marken 57
   3.4. Verzweigungen 59
      3.4.1. Unbedingte Sprünge 61
      3.4.2. Bedingte Sprünge 63
      3.4.3. Schleifen 69
   3.5. Unterprogramme 76

4. Das Programm ... 81
4.1. Problemanalyse ... 83
4.2. Ablaufschema ... 86
4.3. Festlegen der Programmschritte ... 89
4.3.1. Beispiel in der algebraischen Logik ... 90
4.3.2. Beispiel in der umgekehrten polnischen Notation ... 92
4.4. Programmverarbeitung ... 93
4.4.1. Programmeingabe ... 93
4.4.2. Test ... 94
4.4.3. Programmspeicherung ... 96
4.4.4. Fehlerdiagnose ... 97
4.4.5. Redigierung ... 98
4.5. Programmdokumentation ... 99
4.5.1. Bedienungsanleitung ... 99
4.5.2. Programm–Listung ... 100
4.5.3. Programmbeschreibung ... 101
4.5.4. Testbeispiele ... 101
4.6. Änderungsdienst ... 102
4.7. Hinweise für Fortgeschrittene ... 102
4.7.1. Mehrfachbelegung von Speichern ... 103
4.7.2. Datenorganisation ... 106
4.7.3. Mehrfachverwendung einer freien Funktionstaste ... 107
4.7.4. Eingabe während einer Pause ... 108
4.7.5. Flag–Logik ... 109
4.7.6. Rechenzeit ... 110

5. Formblattwesen ... 112

6. Anwendungen ... 114
6.1. Einsatzmöglichkeiten von Kleinrechnern ... 115
6.2. PROTAR und Großrechenanlage ... 115
6.3. Organisation einer Programmierung ... 115
6.4. Anwendungsprogramme ... 116
6.4.1. Standardprogramme ... 116
6.4.2. Individualprogramme ... 116

7. Auswahlkriterien ... 118

Nachwort ... 120

Beispielsammlung

Beispiel A: Industrielle Kalkulation ... 121
B: Industrielle Kalkulation automatisch ... 126
C: Mengenauflösung ... 129
D: Kalkulatorische Kosten ... 131
E: Kalkulatorische Kosten automatisch ... 134
F: Auftragsbearbeitung ... 136
G: Energieverbrauchskontrollen ... 138
H: Auswertung von Zeitaufnahmen ... 142
I: Tabellenaufrechnung (mit kompletten Programmunterlagen) ... 144
K: Güteprüfung von Bordstelnen ... 150

Sachwortverzeichnis 152

1. Einleitung

Warum eigentlich programmierbare Taschenrechner? Sollte man nicht vorerst einmal den TAR verdauen, ehe man sich dem PROTAR zuwendet? Ist ein programmierbarer Taschenrechner nicht ein Produkt eines Größenwahns, es mit untauglichen Mitteln größeren Rechenanlagen gleich tun zu wollen? Es ist mir nicht bekannt, welche Gründe für die Entwicklung von PROTARen bei den einzelnen Firmen maßgebend waren. Sicherlich mag ein Grund im besonderen eine gewisse Bedeutung dabei gehabt haben, nämlich der, daß beim Bau der Elemente von Rechenanlagen mengenmäßig recht großzügig vorgegangen wurde und daher diese Elemente durch die Massenfertigung immer billiger geworden sind. So billig, daß man begann, damit kleine und kleinste Schaltkreise zu bauen und erkannte, was man dabei entdeckt hat: nämlich die Möglichkeit, Rechner zu bauen, die im Prinzip wie ein größerer Computer funktionierten, der nach einmal vorgegebenen Rechenanweisungen einen Vorgang immer wieder automatisch ablaufen läßt.

Und es gibt auch viele Gründe, die für einen Einsatz von PROTAR sprechen. Ein Techniker wird aufgrund seiner mathematischen Begabung selbst bald zum PROTAR finden. Er sieht dessen Anwendungsmöglichkeiten beispielsweise in Iterationsrechnungen. Das sind Rechenvorgänge, die immer wieder ablaufen müssen, bis das Ergebnis die gewünschte Genauigkeit aufweist. Diese Näherungsrechnung empfiehlt sich vor allem bei der Berechnung komplizierter Formeln. Oft ist eine wiederholte Durchrechnung von Hand überhaupt nicht zu bewältigen und wenn, dann nur unter großen Mühen und Zeitaufwand. Außerdem ist bei dieser Vorgangsweise mit einer entsprechend hohen Fehlerquote zu rechnen. Ein lohnender Einsatz für PROTAR! Ein Statistiker wird sich eines programmierbaren Rechners bedienen, um komplizierte Analysen zu erstellen, und wenn sie nicht übergroß sind, kann er sie auch auf einem PROTAR bewältigen. Denken Sie beispielsweise an die statistische Auswertung von Prüfergebnissen, die auch mit PROTAR spielend zu bewältigen ist.

Es ist aber der im kaufmännischen und betriebswirtschaftlichen Bereich Tätige, an den sich unsere Ausführungen im besonderen richten; denn er hat es doch etwas schwerer als der Techniker, mit komplexeren Problemen fertig zu werden. Gerade im genannten Arbeitsbereich hat man es aber mit relativ einfachen Rechenvorgängen zu tun, die den für PROTAR unschätzbaren Vorteil haben, sehr häufig vorzukommen und vor allem nicht allzu lang zu sein. Dies trifft beispielsweise auf eine Kalkulation zu, die immer wieder nach dem gleichen Schema abläuft. Auch eine kleine Anlagenbuchhaltung zur Ermittlung der kalkulatorischen Kosten ist für PROTAR ein reines Vergnügen!

Wie erwähnt, sind im kaufmännischen Bereich die rein mathematischen Probleme weniger kompliziert, als es in der Mathematik beispielsweise die Berechnung eines Integrals ist. Das Fluidum des schwer Überwindlichen, das PROTAR für manche umgibt, soll im Rahmen unserer Ausführungen etwas gelüftet und dem potentiellen Benützer, aus welchem Bereich er auch kommen möge, ein wenig an die Hand gegangen werden.

Selbst habe ich es mehrfach durchexerziert: Ein Detailproblem geringen Umfanges wurde im Ablauf auf dem kleinstmöglichen Rechner erprobt, um es dann auf einem größeren Rechner in ein größeres Programm einbauen zu können. Je größer der Rechner, um so größer ist auch Vorarbeit und Testaufwand. Im besonderen denke ich an ein Beispiel, das in den späteren Ausführungen sowie in der Beispielsammlung noch ausführlich diskutiert wird. Es geht um das Problem, drei Zahlen der Größe nach einzuordnen und beim Auftauchen einer neuen Zahl abzuprüfen, ob sie in diese Reihe hineinpaßt oder nicht.

Dieses Problem stellt sich beispielsweise bei einer industriellen Stromabrechnung. Der zur Verrechnung gelangende Grundpreis orientiert sich an der bezogenen Leistung in KW. Und zwar wird dieser Verrechnungswert aus dem arithmetischen Mittel der drei höchsten Monatslasten eines Rechnungsjahres ermittelt. Im Zuge der Berechnung der Stromkosten

eines Monats wird das neue Monatsmaximum mit den drei bereits vorhandenen Höchstlasten verglichen und allenfalls in diese Reihe eingeordnet, wenn zumindest der bisherige kleinste Höchstwert kleiner als die neue Höchstlast ist. Aus diesen drei neuen Höchstlasten wird erneut ein arithmetisches Mittel errechnet, das dann der Ermittlung des Grundpreises zugrunde gelegt wird. Das klingt hier komplizierter als es tatsächlich ist. Im Zuge unserer Ausführungen werden wir noch auf dieses Beispiel zurückkommen.

Zu Beginn dieser Einleitung haben wir von Schaltkreisen gehört, aus denen ein Rechner in seinen wesentlichsten Bestandteilen gebaut ist. Es seien dabei auch die „gedanklichen" Schaltkreise nicht vergessen, mit denen auf den zentralen Punkt in der Programmierung schon an dieser Stelle verwiesen werden soll. Diese gedanklichen Vorarbeiten zur Erstellung eines Programms bestehen im Erkennen der Zusammenhänge der Elemente eines Problems. Der Eingeweihte weiß, daß verschiedene Firmen – nicht nur die Hersteller elektronischer Rechenanlagen – Stellenbewerber logischen Tests unterziehen. Es handelt sich dabei um Aufgaben, mit denen das logische Denkvermögen im wesentlichen überprüft werden soll.

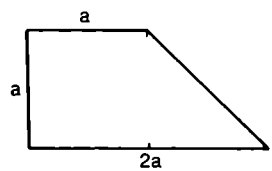
Sie als Leser sollen nun nicht einem derartigen Test unterworfen werden; doch seien einige Beispiele herausgegriffen, um an diesen klar zu machen zu versuchen, worum es in diesem Buch im wesentlichen geht. Dazu möchte ich Ihnen vier Zahlenbeispiele und ein geometrisches Beispiel vorstellen und Sie bitten zu versuchen, diese zu lösen. Die Ergebnisse Ihrer Bemühungen können Sie dann mit den am Ende dieser Einleitung plazierten Lösungen vergleichen. Wir werden dort auch miteinander die Lösungswege diskutieren und uns mit diesen auseinandersetzen.

Wir haben es mit jeweils zwei Zahlengruppen zu tun. Die erste Gruppe ist mit einer Zahl aus der zweiten Gruppe zu ergänzen, und zwar an der Stelle des Fragezeichens.

1. 1,3,5,7,9, ?	8,10,11,13,14
2. 2,4,3,5,4, ?	3,5,6,7,9
3. 2,4,8,16,32 ?	48,54,62,64,74
4. 4,5,10,11,22 ?	23,24,26,28,33

Wenn Sie nun diese Fragen geklärt haben, notieren Sie sich bitte vorerst einmal die Ergebnisse, zu denen Sie gekommen sind und versuchen das fünfte Beispiel zu treffen, bevor Sie Ihre Ergebnisse mit den Lösungen am Ende dieses Kapitels vergleichen.

5. Nachstehende Figur ist in vier deckungsgleiche Flächen zu unterteilen.



Der Leser fühle sich nicht aufs Eis geführt; er versuche nur, hier Abhänglgkeiten und Zusammenhänge zu erkennen und die „logischen" oder „gedanklichen" Schaltkreise, die in den Problemen ihre Wurzeln haben, klar zu legen.

Unterläuft einem bei der Programmierung eines Großrechners oder auch eines Bürocomputers der sogenannten mittleren Datentechnik einmal ein Irrtum, ist im allgemeinen die Fehlerursache schwer zu lokalisieren, da die Wirkung des Fehlers oft erst an ganz anderer Stelle im Programm offenbar wird; dies gilt zwar auch im Hinblick auf unseren PROTAR,

nur ist bei ihm die Suche und vor allem die Korrektur des Fehlers wesentlich weniger aufwendig; wieder ein Argument mehr für PROTAR. Wie weit die Möglichkeiten gehen, soll ein weiteres Beispiel zeigen.

Dem Kostenrechner ist der Begriff „Kalkulatorische Kosten" geläufig. Es handelt sich hier um die von der steuerlichen Behandlung unabhängig ermittelte Abschreibung für Abnutzung und um die Zinsen von Anlagegütern eines Unternehmens. Auf einem Taschenrechner wurde ein Programm zur Ermittlung dieser Kosten entwickelt, das eine Art Anlagenbuchhaltung darstellt. Daraufhin konnte man es als Vorlage zur Erstellung des gleichen auf einem Magnetkontencomputer ablaufenden Programms benutzen.

PROTAR kann also verwendet werden

- als normaler Taschenrechner;
- als Hilfsmittel zur Lösung von Detailproblemen, die dann in Programme größerer Anlagen eingebaut werden;
- zur Lösung der vielen kleinen täglich anfallenden Probleme.

Und diesem letzten der drei Möglichkeiten wollen wir uns ganz besonders widmen. Sie mögen im Verlaufe dieser Ausführungen bei sorgsamer Durcharbeitung sicherlich entdekken, wieviel Eigenleben PROTAR bei Ihnen entwickeln wird. Falls Sie doch noch skeptisch sind, was PROTARe zu leisten imstande sind, nehmen Sie als erstes das Kapitel der Anwendungen in Angriff. Sie werden zwar bei Details Schwierigkeiten bekommen, aber zumindest schon die Absicht erkennen, die in diese Beispiele hineingelegt wurde und merken, wieviele Aufgaben sich mit PROTARen lösen lassen. Grundsätzlich meine ich, daß sich die Fähigkeit zu analytischem Denken – an sich eine wesentliche Voraussetzung für das Erkennen und Lösen von Problemen bis zur Anwendung der Problemlösungen mit PROTAR – schulen läßt. Darüber hinaus haben wir es ja mit verhältnismäßig leicht überschaubaren und kleinen Problemen zu tun. Die schwierigeren Fälle und jene, die sehr komplex und vor allem lang sind, müssen ohnehin mit Hilfe größerer Rechenanlagen abgewickelt werden. Was wir hier tun müssen, ist leicht getan: Erkennen, wo häufige Wiederholungen auftreten und Beobachten dieser Abläufe und Umsetzen dieser Erkenntnisse in Programme.

Der große Vorteil an PROTAR ist, daß sein Herr Analytiker, Programmierer und Operator **in einer Person** ist. Er zerlegt die Probleme in leicht kontrollierbare Einzelteile, schreibt das Programm und benutzt es auch. Er erkennt auch die Problematik seiner Arbeit selbst am besten und wird daher – was die Arbeitsabläufe betrifft – auch die geeignetsten Programme schreiben, wenn diese Programme auch programmtechnische Mängel aufweisen mögen. Man könnte einwenden, das ist ja alles recht schön und gut und ein Programm für eine bestimmtes Problem müßte ja auch die Lieferfirma Ihres PROTARs vorrätig haben. An dieser Stelle werden Sie nun schon die ersten Schwierigkeiten in der Erlangung von Hilfe seitens der Hersteller bekommen. Man wird Sie in einem höflichen Antwortschreiben unter Beifügung eines zusätzlichen Kommentars bestenfalls an die jeweilige Bedienungsanleitung verweisen. Denn Hand auf's Herz: Wer wird bei den Preisen, zu denen PROTARe heute gehandelt werden, auch noch ausführliche Programmunterstützung verlangen können? Dazu kommt noch, daß Sie Ihren PROTAR vor allem über Handelsketten und Märkte beziehen müssen. Sicherlich wird Ihnen Ihr Lieferant Standardprogramme anbieten können, die aber – abgesehen von rein technischen Problemlösungen – nicht immer in allen Punkten Ihren Vorstellungen entsprechen werden. Vor allem auch deswegen, weil PROTARe aufgrund ihrer Konzeption wie geschaffen zur Lösung Ihrer individuellen Probleme sind. Deswegen müssen Sie da auch ein wenig für Ihre persönliche Information tun.

In diesem Sinne darf ich Sie nun bitten, mir durch die nachfolgenden Kapitel zu folgen.

Lösungen der Beispiele

Lösung Beispiel 1: 11.

Begründung: Wir erkennen folgende Gesetzmäßigkeit: Jede nachfolgende Zahl unterscheidet sich von der vergangenen um den Wert 2, sodaß nach 9 nur 11 kommen kann.

Lösung Beispiel 2: 6.

Begründung: Die hierin enthaltene Gesetzmäßigkeit ist zweiteilig; die Unterscheidung einer nachfolgenden von der vorangegangenen Zahl wechselt zwischen +2 und −1. Nach 4 Differenzen war wieder +2 an der Reihe, sodaß nach 4 nur 6 kommen konnte.

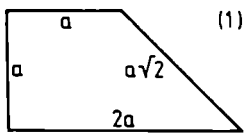
Lösung Beispiel 3: 64.

Begründung: Hier haben wir es wieder mit einer einteiligen Gesetzmäßigkeit zu tun. Jede nachfolgende Zahl ist doppelt so groß, wie die vorangegangene, sodaß nach der Zahl 32 nur 64 stehen kann.

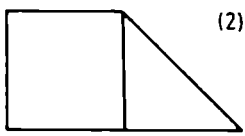
Lösung Beispiel 4: 23.

Begründung: Die nun zweiteilige Gesetzmäßigkeit ergibt hier einen Wechsel zwischen +1 und ×2. Der 5. Unterschied muß daher +1 sein, d. h. die 6. Zahl konnte nur 23 lauten.

Lösung Beispiel 5: Wir müssen versuchen

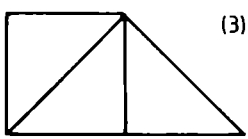


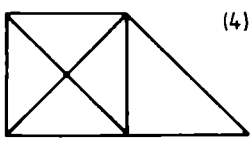
(1)


(2)

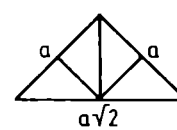
a) die Figur in 12 gleiche Teile zu zerlegen. 12 ist das kleinste gemeinsame Vielfache von der geforderten Teilzahl 4 und der in (3) offensichtlich leicht erkennbaren Teilbarkeit durch 3.

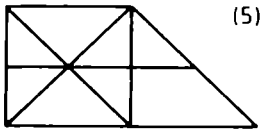
b) zu berücksichtigen, daß auch die Lösungsfigur eine schräge Seite aufweisen muß.


(3)


(4)

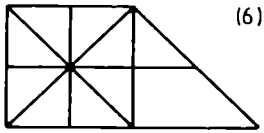
Hier haben wir nun das gegebene Trapez in den drei Grundfiguren, das sind gleichschenkelig rechtwinkelige Dreiecke mit den Seiten a, a und a√2. Dabei ist a√2 die jeweilige Grundlinie. Mit den weiteren Schritten müssen wir daher jedes dieser drei Dreiecke in vier ähnliche Dreiecke zerlegen. Dies dürfte nun kaum mehr Schwierigkeiten bereiten.



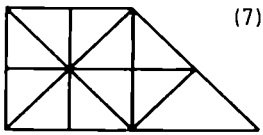


(5)

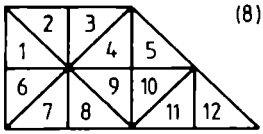
Wenn Sie genau hinsehen, erkennen Sie vielleicht jetzt schon die Lösungsfigur!



(6)



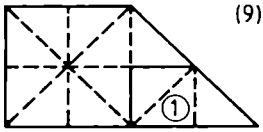
(7)



(8)

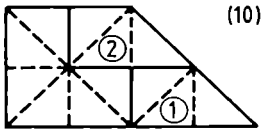
Mit diesem letzten Teilstrich haben wir nun das Trapez in 12 gleichschenkelig–rechtwinkelige Dreiecke zerlegt.

Wir müssen nun Gruppen zu drei Dreiecken suchen, um der Forderung nach Vierteilung des ursprünglichen Trapezes gerecht zu werden und auch dabei eine schräge Seite in die Lösung mit hineinzubekommen.

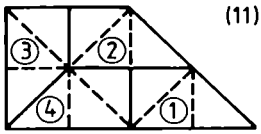


(9)

Wir beginnen am äußersten Eck und zählen die ersten drei Dreiecke heraus; die einzig mögliche Kombination! Wir erkennen darin wieder unsere Ausgangsfigur, nur wesentlich verkleinert. Nun nehmen wir die nächste noch verfügbare schräge Seite und gewinnen damit die zweite der vier Lösungsfiguren.



(10)



(11)

Nun kann mit einem Strich der Rest geteilt werden. Dabei merken wir, daß nun eine der vier Lösungsfiguren seitenverkehrt ist. Die ursprüngliche Forderung bestand aber nur in der Suche nach vier deckungsgleichen Figuren, wobei eine seitenverkehrte Lösung nicht ausdrücklich ausgeschlossen war.

Die Darstellung dieser Probleme und ihrer Lösungen und der darin enthaltenen Zusammenhänge soll uns zeigen, worauf es letztlich auch in der Programmierung ankommt. Eine vorgezogene Schulung des Vermögens zum analytischen Denken, ohne die wir im folgenden nicht ganz auskommen können.

2. Der Rechner

Was man heute an Funktionen und Möglichkeiten in einen Rechner hineinpacken kann, auf immer kleiner werdende Flächen unterbringt und in immer knapper werdende Räume zu verstauen versteht, grenzt wahrhaft schon an Zauberei. Man braucht nur die einzelnen Verstärkerelemente in der Reihenfolge ihrer Entwicklung betrachten:

- Röhre;
- Transistor;
- gedruckte Schaltung.

Diese Dinge sind nicht allein auf die Rechentechnik beschränkt, sie werden überall dort verwendet, wo elektrische Schaltungen montiert werden müssen. Stark vergröbert ausgedrückt war die Weltraumfahrt ohne den Einsatz von Computern unvorstellbar und undurchführbar. Durch die Massenerzeugung wurden diese Schaltelemente immer billiger und davon profitieren vor allem wir mit unseren PROTARen. Aber in nahezu jeder elektrischen Schaltung bedient man sich heute der in der Hauptsache aus Silicium bestehenden „IC's" – der „integrated circuits". Auf einem Minimum an Fläche kann ein Maximum an Funktionen untergebracht werden. Nehmen Sie einen einfachen Rechner in die Hand und Sie werden merken, daß Sie gewichtsmäßig kaum etwas in der Hand haben, PROTARe sind da kaum schwerer.

Wir haben schon gehört, daß die PROTARe zur großen Familie der Taschenrechner gehören. Sie funktionieren wie jeder normale Rechner. Sie können aber auch – und das unterscheidet sie von den übrigen „Familienmitgliedern" wesentlich – nach vorgegebenen Rechenvorschriften, den Programmen, die Vorgänge im Rechner automatisch ablaufen lassen, und das nicht nur einmal, sondern immer wieder. Ob man nun programmierbare Taschenrechner, bei denen eine externe Speicherung dieser Rechenvorschriften nicht möglich ist, deswegen nicht mehr als Computer bezeichnen kann im Gegensatz zum fast gleichen Modell, wo dies der Fall ist, halte ich im Gegensatz zu Osterloh für eine zu akademische Erörterung. Das entscheidende Merkmal für einen Computer ist für mich in erster Linie der **automatisch ablaufende Rechenvorgang**. Ob Sie nun PROTAR als einen Computer in Mini–Ausführung bezeichnen wollen oder nicht, wesentlicher als die Lösung dieser Frage scheint mir wohl die Vielfalt seiner Nutzungsmöglichkeiten zu sein.

Sie finden in einem späteren Kapitel auch Hinweise, nach denen ich bei einem gegebenen Problem einen programmierbaren Taschenrechner auswählen würde. Diese Hinweise verstehen sich als Entscheidungshilfe beim Kauf eines Rechners; wir aber wollen unser Hauptaugenmerk auf eine optimale und wirtschaftliche Lösung unserer Probleme mit PROTAR legen.

Der Schwierigkeit bewußt, einen dem heutigen Stand der Technik entsprechenden Überblick zu geben, werde ich trotzdem versuchen, da und dort auf verschiedene Details einzugehen. Das Hauptanliegen meiner Ausführungen ist da zu suchen, wo ich immer wieder auf den Begriff „Problem" und seine Darstellung eingehe und wo Ich bei Ihnen ein wenig Verständnis dafür zu wecken versuche, wie PROTARe in der Lage sein können, ein wenig mehr Licht in Ihre Tagesarbeit zu bringen.

Sie sollen dadurch noch besser in die Lage versetzt werden, Ihre ureigensten Probleme zu erkennen und formulieren zu lernen um zu sehen, welchen Stellenwert PROTARe heute auch für Ihren Bedarf einnehmen. Daher erscheint es mir auch wichtig, zuvor ein wenig mit dem Rechner vertraut zu werden, vor allem dann, wenn Sie bis dato nicht einmal Besitzer eines TARs waren. Je besser Sie Funktionen und Zusammenhänge verstehen, desto besser sind Sie später imstande, PROTAR Ihren Diensten zu unterwerfen.

2.1. Aufbau eines Rechners

Ein Rechner besteht im allgemeinen aus einem

- **Tastenfeld** (Eingabevorrichtung), der
- **Rechenlogik** (nicht sichtbares Rechensystem), der
- **Anzeige oder Druckeinrichtung** (Ausgabevorrichtung), dem
- **Speicher** zur Aufbewahrung von Programmen und Daten, der
- **Energieversorgung** durch Batterien und Netz und dem
- **Gehäuse**, das alles das zusammenhält.

Über das **Tastenfeld** werden dem Rechner die Aufgaben gestellt, die zu lösen sind. Wir geben dem Rechner durch Betätigung verschiedener Tasten einen Auftrag, der dann in kaum meßbarer Zeit ausgeführt wird. Diesen Auftrag, bzw. diese Reihe von Tastendrücken oder auch **Befehle** genannt bezeichnet man in der Programmsprache als **Sequenz** oder **Befehlsfolge**. Mit Hilfe der **Rechenlogik** wird dann das gestellte Rechenproblem gelöst und über die **Anzeige** oder auch **Druckeinrichtung** gibt Ihnen der Rechner das Ergebnis seiner Bemühungen bekannt. Und dann werden Sie staunen, daß er zu einem anderen Ergebnis kommt, als Sie es erwartet haben. Denn als schlauer Fuchs haben Sie das Problem vorher zu Fuß gelöst und wollten nun PROTAR „auf's Kreuz" legen. Ehrlich gesagt, ich würde in diesem Fall doch lieber nochmals in der Bedienungsanleitung nachsehen, ob die Tasten denn auch in der richtigen Reihenfolge gedrückt worden sind. Denn da muß ich meinen Freund PROTAR doch ein wenig in Schutz nehmen: In den meisten Fällen liegt wohl ein Bedienungsfehler vor, denn PROTAR macht alles das, was Sie ihm auftragen und das mit einerAkkuratesse, die ihresgleichen sucht. Im Falle, daß Sie ihm doch mißtrauen, haben die Hersteller Diagnoseprogramme mitgeliefert, mit deren Hilfe Sie die Funktionstüchtigkeit Ihres PROTARs im wesentlichen überprüfen können. Denn Mängel sind sicherlich nicht auszuschließen. Erst wenn diese Diagnosetests positiv verlaufen, also Funktionsmängel zu Tage förderten, müssen Sie den Rechner dem Hersteller zur Überprüfung einsenden. Aus diesem Grunde empfiehlt es sich, während der Garantiezeit Ihrem PROTAR alles abzuverlangen und das auch sehr oft. Mängel in den Schaltkreisen und in der Mechanik zeigen sich im allgemeinen bald. Lassen Sie Ihren Rechner mitunter auch lange Zeit in Tätigkeit, damit unter Wärmeeinwirkung unter Umständen gewisse Fehler leichter auftreten.

In diesem Zusammenhang wollen wir auch noch etwas näher auf die **Energieversorgung** eingehen. Reine batteriebetriebene Geräte sind mir bei programmierbaren Taschenrechnern nicht bekannt. Es wäre auch unwirtschaftlich, PROTARe mit Batterien zu betreiben. Sollte es solche Geräte aber tatsächlich geben, ist vor einem Kauf abzuraten.

Alle mir bekannten Modelle sind also sowohl über wieder aufladbare Batterien, als auch mit Hilfe mitgelieferter Netzanschlüsse, zu betreiben. Bei älteren Ausführungen funktioniert die Energieversorgung über die Batterien nur dann, wenn das Netzkabel mit Ladeeinheit vom Rechner getrennt war.

Heute gilt diese Einschränkung im allgemeinen nicht mehr. Ist das Netzkabel angeschlossen, kann sowohl mit Batterien als auch mit dem Netz gearbeitet werden. Das hat einen unschätzbaren Vorteil: Sie merken einen Stromausfall nicht. Der Rechner schaltet bei Unterbrechung der Stromzufur vom Netz automatisch auf Batteriebetrieb um, ohne daß Sie es merken und können daher ohne Unterbrechung und vor allem ohne Schaden weiterarbeiten. Denn eines können PROTARe bis auf wenige Ausnahmen (s. S. 120/5) nicht: Ihr „Gedächtnis" behalten, wenn die Energiezufuhr unterbrochen wird. Speziell bei einer länger andauernden Verarbeitung ist die Sicherheit von größter Wichtigkeit. Innerhalb der Entleerdauer der Batterien Ihres PROTARs kann im allgemeinen damit gerechnet werden, daß der Netzstrom wieder zur Verfügung steht. Sollte auch dies nicht genügen, müßten Sie eben bei Vorhandensein eines Modells mit externer Speichermöglichkeit von Programmen

und Daten vor endgültigem Energieverlust die Daten auf einem Magnetstreifen aufzeichnen und damit sichern; **Datensicherung** auch im Bereich der PROTARe, ein völlig neuer Aspekt! Sie können nun getrost nach der so vorbereiteten Unterbrechung bei Wiederherstellung der Stromversorgung weiterarbeiten. Sie müssen vorher nur sehen, daß Sie noch bis zu einem definierten Programmstop kommen, von wo Sie dann anschließend fortfahren können. Bei Unterbrechungen mitten in einer Verarbeitung heißt es allerdings leider meist, von vorne beginnen. Der Zustand der Batterien ist aber soweit beurteilbar, daß Sie noch genügend Zeit zur Datensicherung haben, vorausgesetzt, Sie betreiben einen PROTAR mit externer Speichermöglichkeit von Programmen und Daten. Eine derartige Unterbrechung kann aber auch aus anderern Quellen herrühren. Sie bekommen mitten in Ihrer Arbeit Besuch von lange nicht mehr gesehenen Freunden, die Sie auf ein „kühles Blondes" oder einen Schoppen Wein einladen wollen. Vor lauter Wiedersehensfreude ist es dann passiert: Sie haben den Rechner abgeschaltet oder wenn nicht, hat doch irgend ein Spitzbube während Ihrer Abwesenheit am PROTAR herumgespielt, was letzten Endes den gleichen „Erfolg " hatte: Sie können wieder von vorne beginnen. Wären Sie wie zuvor beschrieben vorgegangen, hätten Sie das Programm bis zu einem Programmstop weiterarbeiten lassen und anschließend die Daten abgespeichert, hätten Sie sich bei Wiederinbetriebnahme des Programms viel Zeit sparen können. Dann wäre auch das Wiedersehen in besserer Erinnerung; also: Programm bis zu einem erkennbaren Stop fortsetzen, die Daten auf einen Magnetstreifen abspeichern, Magnetstreifen kennzeichnen, PROTAR abschalten und wegstellen – und dann die Wiedersehensfeier voll auf sich einwirken lassen! Das gleiche gilt selbstredend auch für die unvermeidliche Unterbrechung durch den Arbeitsschluß am Abend eines Arbeitstages. Sie wissen dann, daß Sie am nächsten Tag dort fortsetzen können, wo Sie am Tag zuvor unterbrechen mußten und das ohne Komplikationen. Das kann z. B. bei Exekution einer Anlagenbuchhaltung der Fall sein, eine sich sicher über einen längeren Zeitraum erstreckende Arbeit.

Datensicherung können Sie auch bei kleineren Geräten betreiben, einfach damit, daß Sie vor einer Unterbrechung die Inhalte der Datenspeicher abrufen und aufnotieren. Vor Wiederaufnahme der Verarbeitung sind die Speicherinhalte dem Rechner wieder einzugeben. Die Möglichkeiten von Unterbrechungen sind bei solchen Geräten allerdings kaum gegeben, denn mit zunehmender Kleinheit der Geräte werden auch die bearbeitbaren Probleme kleiner, sodaß man kaum mit der zuvor beschriebenen Situation konfrontiert werden dürfte. Bei einigen kleinen PROTARen eines Herstellers (s. S. 120/5) gibt es hierfür eine Abhilfe: Bei geringstem Stromverbrauch wird die Magnetisierung der Programm- und Datenspeicher zumindest so lange aufrecht erhalten, bis Sie Ihr kühles Blondes oder die dazwischen liegende Nacht hinter sich haben und weiterarbeiten, als ob inzwischen nichts geschehen wäre. Diese Eigenschaft bei kleinen PROTARen wurde allerdings zu einem anderen Zweck eingerichtet, um für Ein–Zweck–Verwendungen das Programm nicht immer eingeben zu müssen; denn diese Modelle sind im allgemeinen nicht mit externer Speichermöglichkeit von Programmen und Daten ausgestattet. Man sieht, daß schon bei Osterloh diskutierte Probleme auch hier auftauchen. Wir wollen uns aber im speziellen mit PROTARen befassen und da läßt es sich nicht vermeiden, daß auf bestimmte Dinge auch hier der Vollständigkeit halber erneut eingegangen werden muß; ich hoffe hier auf Verständnis.

Das gleiche gilt auch im speziellen Fall, wenn wir so einfache Dinge, wie das Tastenfeld, besprechen. Denn ich muß davon ausgehen, daß die vorliegende Einführung in die Programmierung von Taschenrechnern vor dem Kauf eines PROTARs studiert wird, oder besser studiert werden sollte. Sie sollen hier nur einen ersten Eindruck vom Tastenfeld bekommen. Durch die Ausführungen in den Bedienungsanleitungen soll dieser erste Eindruck dann erweitert und vertieft werden.

2.2. Rechenlogik

Was heißt Rechenlogik und was bedeutet sie? Einerseits ist diese Rechenlogik nichts anderes als eine rechnerintern vorgeschriebene Reihenfolge zur Lösung bestimmter Rechenoperationen. Zahlentasten und Funktionstasten sind nach bestimmten Vorschriften zu betätigen. Andrerseits hat die Rechenlogik im weiteren Sinne auch die Aufgabe, die Vorgänge bei automatischem Ablauf zu steuern. Es spielt so ein wenig die Rolle eines Betriebssystems. Diesen Begriff kennen wir aus der DV. Er bedeutet nichts anderes als ein Steuerprogramm, das vorhanden sein muß, um der Ansammlung von Schaltkreisen, die in einem Gehäuse verpackt sind, Leben einzuhauchen. Und diese Funktion übernimmt bis zu einem gewissen Grad auch die Rechenlogik. Ein Betriebssystem ist also eine recheninterne Steuerung. Bei PROTARen ist sie bereits eingebaut.

Diese Rechenlogik spiegelt sich naturgemäß im Aufbau der Bedienungsanleitung wider. Schon im Hinblick auf die unterschiedliche Ausstattung mit Funktionstasten ist einzusehen, daß man sich an jedes Gerät neu gewöhnen muß; genau so verhält es sich aber auch, wenn man mit einem Wechsel des Gerätes auch einen Wechsel in der Rechenlogik mitmachen muß.

Im wesentlichen gibt es zwei Logikarten, die wiederum in einigen Spielarten variieren:

- Die algebraische Logik, im folgenden mit AL bezeichnet und
- die umgekehrte polnische Notation, im folgenden mit UPN abgekürzt.

Wir wollen nun die einzelnen Möglichkeiten näher kennenlernen und die wesentlichen Unterschiede herauszuarbeiten versuchen.

2.2.1. Die algebraische Logik

Die AL wird auch als die natürliche Eingabemethode bezeichnet, da den Rechnern, die damit ausgestattet sind, die Rechenoperationen genau in der Reihenfolge einzugeben sind, wie sie bei herkömmlichen schriftlichen Rechenarbeiten auf dem Papier niedergeschrieben werden. Der erste Operand wird eingegeben, anschließend eine der vier Grundfunktionstasten gedrückt. Hierauf erfolgt die Eingabe des zweiten Operanden, wonach die Rechnung mit dem Drücken der Gleichheitstaste abgeschlossen wird.

Beispiele:	12	+	9	=;	Anzeige 21 oder	100	−	22	=;	Anzeige 78 oder
	2,3	×	5	=;	Anzeige 11,5 oder	45	÷	15	=;	Anzeige 3.

Man sieht, daß die Bilder und Abläufe genau denen gleichen, die uns seit dem Schulunterricht vertraut sind. Wir haben die **algebraische Reihenfolge** benutzt. Daher sind die mit diesem System ausgestatteten Rechner besonders leicht zu bedienen und werden auch häufig im Schulunterricht eingesetzt, da sie dem allgemeinen üblichen und natürlichen Rechenablauf folgen.

Wir haben auch gesehen, daß die Gleichheitstaste den Rechenablauf beendet, an sich eine Selbstverständlichkeit. Diese Taste ist ein Hinweis darauf, daß dieser PROTAR mit der AL ausgestattet ist. Daher wollen wir uns einmal näher ansehen, was bei den einzelnen Schritten passiert, die wir in den trivalen Beispielen kurz zuvor erwähnt haben:

- Eingabe des ersten Operanden: Die erste Zahl wird eingegeben und befindet sich im Eingabespeicher.
- Drücken einer Funktionstaste: Die eingegebene Zahl wird in das Rechenregister übertragen, verbleibt aber gleichzeitig in der Anzeige. Die Zahl zeigt jetzt, wenn es sich bei der ersten Eingabe um eine ganze Zahl gehandelt hat, rechts von der Einerstelle automatisch den Dezimalpunkt an, was vorher nicht immer der Fall gewesen sein muß. Damit wird angedeutet, daß der Eingabespeicher zur Eingabe des zweiten Operanden, der zweiten Zahl, frei gegeben ist.

- Eingabe des zweiten Operanden: Die zweite Zahl wird in den Eingabespeicher eingegeben und – falls es eine ganze Zahl ist – unter Umständen auch ohne Dezimalpunkt angezeigt.
- Drücken der Gleichheitstaste: Damit wird einerseits die Zahleneingabe abgeschlossen und andererseits die angewählte Funktion ausgelöst, die Rechnung ausgeführt. Das Ergebnis erscheint in der Anzeige. Bei PROTARen mit Drucker wäre, um das Ergebnis angedruckt zu erhalten, der Druckbefehl von Hand aus zu betätigen. In einem Programm kann durch Befehlsvorwahl diese Funktion automatisch ausgelöst werden.

Wie wir auch noch vom Schulunterricht wissen, gelten hinsichtlich der Bearbeitung einer Kettenrechnung eigene Regeln. Multiplikationen und Divisionen sind Operationen einer höheren Ordnung, als Additionen und Subtraktionen. Um hier eine entsprechende Reihenfolge gewährleisten zu können, sind bei bewußter Abweichung von dieser Ordnung diese Abweichungen durch Setzen von Klammern entsprechend zu kennzeichnen. Wir wollen das wieder an einem Beispiel demonstrieren:

$2 + 3 \cdot 6 - 4 \cdot 2 =$; Anzeige 12.

Diese Rechnung soll nun umgestaltet werden:

$2 + 3 \cdot (6 - 4) \cdot 2 =$; Anzeige 14.

Ein Versetzen der zweiten Klammer bringt folgende Veränderung des Ergebnisses:

$2 + 3 \cdot (6 - 4.2) =$; Anzeige – 4.

Sie sehen damit, wie durch das Setzen von Klammern das Ergebnis beeinflußt werden kann. Bei dem folgenden Beispiel müssen Sie Zähler und Nenner des Bruchs in Klammern setzen:

$$\frac{(3 + 3 \cdot 7)}{(5 \cdot 2 - 2)} =; \quad \text{Anzeige 3.}$$

Wir haben erkannt, daß das Setzen der Klammern bei Berechnung von Brüchen notwendige Voraussetzung ist, vor allem bei der Eingabe des Nenners im letzten Beispiel im Falle des Zählers hätte die Eingabe der Klammern unterbleiben können, aber nur bei diesem Beispiel! Daher sollte man, ist man sich seiner Sache nicht ganz sicher, die Klammern auf alle Fälle setzen!

Wir wollen nun zum besseren Verständnis der Dinge das zweite Beispiel in seinen einzelnen Phasen verfolgen:

1. Schritt: Eingabe der Zahl 2, Anzeige 2;
2. Schritt: Funktionstaste + drücken, Anzeige 2.;
3. Schritt: Eingabe der Zahl 3, Anzeige 3;
4. Schritt: Funktionstaste x drücken, Anzeige 3.;
5. Schritt: Eingabe Klammer auf, Anzeige 3.;
6. Schritt: Eingabe der Zahl 6, Anzeige 6;
7. Schritt: Funktionstaste – drücken, Anzeige 6.;
8. Schritt: Eingabe der Zahl 4, Anzeige 4;
9. Schritt: Eingabe Klammer zu, Anzeige 2., (Zwischenergebnis der Berechnung des Klammerinhalts!);
10. Schritt: Drücken der Funktionstaste x, Anzeige Zwischenergebnis aus 3. (6-4), also 6.;
11. Schritt: Eingabe Zahl 2, Anzeige 2;
12. Schritt: Gleichheitstaste drücken, Anzeige Endergebnis 14.

Wir möchten nochmals darauf hinweisen, daß die Tasten in der gleichen Reihenfolge so zu drücken sind, in welcher Zahlenwerte und Operationszeichen angeschrieben werden. Zusammenfassend kann zur Charakterisierung der AL gesagt werden, daß sie vor allem durch drei Punkte gekennzeichnet ist, und zwar durch das Vorhandensein

1. der Gleichheitstaste,
2. der beiden Klammertasten für „Klammer auf" und „Klammer zu " und der
3. beiden Arbeitsspeicher, wobei nur der eine mit der Anzeige gekoppelte Arbeitsspeicher äußerlich sichtbar ist.

Klammertasten sind allerdings bei einfachen Rechnern, die mit der AL ausgestattet sind, im allgemeinen nicht vorhanden. Daher ist die Gleichheitstaste, d. h. ihr Vorhandensein das erste und auch wichtigste Kennzeichen, das auf die AL schließen läßt.

Im Fall des Fehlens von Klammertasten können Kettenrechnungen in der beschriebenen Form nicht ausgeführt werden. Falls ein Speicher vorhanden ist, sind Zwischenergebnisse darin abzustellen, die dann zu gegebener Zeit wieder abzurufen sind. Oder wir müssen ein Zwischenergebnis als Behelf auf einen Zettel aufschreiben.

In diesem Abschnitt ist es in erster Linie darauf angekommen darzustellen, daß die AL die **natürlichste Form einer Rechenlogik** ist.

Wir haben kennengelernt, was die Merkmale der AL sind und wie man dieses System schon auf große Entfernungen hin identifizieren kann. Es sei hier noch nachzutragen, daß die beiden Arbeitsspeicher mit den Buchstaben x und y gekennzeichnet sind, wobei x gleichzeitig auch die Anzeige darstellt. Die Tatsache des Vorhandenseins der beiden Arbeitsspeicher ist durch die Existenz folgender Tasten auch rein äußerlich erkennbar:

$x \gtrless y$ für den Tausch der Inhalte beider Arbeitsspeicher.

y^x Diese Potenztaste besagt, daß die im Speicher y befindliche Basis mit der in der Anzeige befindlichen Zahl zu potenzieren ist.

Multiplikationszeichen werden bei handschriftlicher Darstellung eines Rechenproblems im allgemeinen nicht gesetzt, wenn Klammerausdrücke miteinander zu multiplizieren sind, zum Beispiel (2 + 3) (2 – 3).

Bei der Eingabe dieses Rechenproblems in den Rechner aber ist das Multiplikationszeichen **unbedingt** zu setzen, da der Rechner sonst nicht weiß, welche Rechenoperation er auszuführen hat. Die Rechnung ist also wie folgt anzuschreiben: (2 + 3) · (2 – 3).

2.2.2. Die umgekehrte polnische Notation

Rechner, die mit UPN arbeiten, sind anfänglich schwerer zu begreifen und zu bedienen. Es gilt aber auch hier das schon mehrfach erwähnte, daß man sich auf den Rechner selbst einstellen muß und dies auch im Bezug auf die Rechenlogik anwendbar ist. Und wie man später noch noch sehen wird, hat auch dieses System seine Vorteile, auch wenn es optisch nicht so natürlich ist, wie die AL. Wir wollen auch hier drei Punkte herausgreifen, weil diese die UPN am ehesten charakterisieren können:

1. Es fehlen Gleichheits- und Klammertasten.
2. Anstelle dieser Tasten haben wir die Tasten ENTER ↑ und in manchen Fällen auch die Stackumspeicherungstasten R ↓ und R ↑ oder zumindest eine der beiden.
3. UPN arbeitet mit vier sogenannten Stackspeichern, das sind vier übereinanderliegende Arbeitsspeicher mit den Bezeichnungen X (auch hier ist der Speicher X gleich der Anzeige), Y, Z und T.

Die mit Hilfe der AL gelösten einfachen Beispiele sollen auch mit UPN in gleicher Weise behandelt werden:

12 ENTER 9 +; Anzeige 21;

2,3 ENTER 5 ×; Anzeige 11,5;

100 ENTER 22 −; Anzeige 78;

45 ENTER 15 ÷; Anzeige 3.

Da wir es mit einem ganz ungewohnten Bild zu tun haben, wollen wir das erste Beispiel genau durchleuchten und uns die Bewegungen in den Arbeitsspeichern näher ansehen. In den einzelnen Arbeitsspeichern mögen vor dieser Rechenoperation bestimmte Werte gespeichert gewesen sein, die wir aber nicht kennen. Zur besseren Leserlichkeit des Beispiels wollen wir für diese uns unbekannten Speicherinhalte Buchstaben wählen und diese in Klammern setzen. Jedem Schritt in der Bearbeitung dieses Rechenproblems wollen wir auch die jeweils letzten Inhalte der vier Arbeitsspeicher voranstellen:

1. Zustand vor der ersten Eingabe:

   T - (D)
   Z - (C)
   Y - (B)
   X - (A)

2. Eingabe des ersten Operanden, der Zahl 12:

   T - (C)
   Z - (B)
   Y - (A)
   X - 12

Die Inhalte sämtlicher Speicher werden um eine Position nach oben verschoben; der Inhalt des T-Speichers geht verloren.

3. Drücken der ENTER-Taste;

   T - (B)
   Z - (A)
   Y - 12
   X - 12

hier erfolgt ein weiterer „Stacklift"; die Eingabe der Zahl wird damit abgeschlossen und die Zahl in den Speicher Y kopiert. Die übrigen Speicherinhalte werden um eine Position verschoben. Damit ist der Eingabespeicher X zur Eingabe der nächsten Zahl frei gegeben, obwohl er im Augenblick noch mit der Zahl 12 belegt ist. Mit anderen Worten, der Inhalt von X wird nach Y, der von Y nach Z und der von Z nach T geschrieben, wobei der ursprüngliche Wert von T verloren geht.

4. Eingabe des zweiten Operanden, der Zahl 9;

   T - (B)
   Z - (A)
   Y - 12
   X - 9

5. Wahl der Funktionstaste +, womit gleichzeitig die Eingabe abgeschlossen und die Rechenoperation ausgeführt wird.

   T - (B)
   Z - (B)
   Y - (A)
   X - 21

Dabei erfolgt die Summation, wobei die Inhalte der Speicher X und Y in X aufaddiert werden. Gleichzeitig erfolgt im Stack das Gegenteil des in 3. beschriebenen Vorganges, nämlich ein sogenannter „Stackdrop": Da Y nach X aufaddiert wird, wandert der Inhalt von Z nach Y, der von T nach Z und der zuletzt in T befindliche Inhalt verbleibt dort und erscheint damit zweimal im Stack, nämlich in T und in Z.

Hier erkennen wir einen kleinen Vorteil der UPN gegenüber der AL: Obwohl gleich viele Tastendrücke zu tätigen sind, wird der Rechenablauf hier durch die Wahl der Funktionstaste beendet. Sie sind nicht schon vor Eingabe des zweiten Operanden an die Funktionstaste gebunden, sondern müssen Ihre Entscheidung erst nach Eingabe des zweiten Operanden treffen; das kann, muß natürlich nicht unbedingt ein Vorteil sein.

Sie können nun die restlichen drei Beispiele der ersten Beispielserie selbst darstellen. Die Beispiele der zweiten Beispielserie würden wie folgt zu bearbeiten sein:

2 ENTER 3 ENTER 6 x ÷ 4 ENTER 2 x –; Anzeige 12,

oder das gleiche Beispiel, aber anders dargestellt:

3 ENTER 6 x 2 + 4 ENTER 2 x –; Anzeige 12,

Sie sehen, daß hier durch Änderungen in der Wahl der Reihenfolge ein Tastenbefehl eingespart werden kann!

Das zweite Beispiel mit dem Ergebnis 14 würde wie folgt ablaufen:

2 ENTER 3 ENTER 6 ENTER 4 – x 2 x + ;

oder anders dargestellt

6 ENTER 4 – 2 x 3 x 2 + ;

Die einzelnen Funktionen lassen einen flexiblen Ablauf zu, wodurch sich gleich zwei Tastenbefehle einsparen lassen.

Im dritten Beispiel sind im Fall UPN folgende Tasten zu drücken:

6 ENTER 4 ENTER 2 x – 3 x 2 + ; Anzeige – 4.

Das vierte Beispiel, die Bruchrechnung mit der Zahl 3 als Ergebnis, läßt sich mit 13 Tastenbefehlen in UPN abwickeln. Die einzelnen der 13 Schritte werden im folgenden gegenüber der Darstellung eines Beispieles in AL nicht textlich, sondern durch die Veränderungen der Inhalte der Stack-Speicher erläutert. Der Einfachheit halber werden hier die Inhalte von X, Y, Z und T nicht untereinander, sondern schrittweise nebeneinander angeordnet.

Schritt-Nr.		Speicher X (A)	Y (B)	Z (C)	T (D)	
1.	3	3	(A)	(B)	(C)	
2.	ENTER	3	3	(A)	(B)	
3.	7	7	3	(A)	(B)	
4.	x	21	(A)	(B)	(B)	
5.	3	3	21	(A)	(B)	
6.	+	24	(A)	(B)	(B)	Zähler ist fertig!
7.	5	5	24	(A)	(B)	
8.	ENTER	5	5	24	(A)	
9.	2	2	5	24	(A)	
10.	x	10	24	(A)	(A)	
11.	2	2	10	24	(A)	
12.	–	8	24	(A)	(A)	Nenner ist fertig! Zähler befindet sich in Y!
13.	÷	3	(A)	(A)	(A)	

Es ist zu beachten, daß sämtliche im X-Speicher eingetragenen Werte in der Anzeige erscheinen, sodaß Sie - im Besitz eines UPN-PROTARs - damit diesen Vorgang ebenso verfolgen können, wie in der Darstellung eines Beispieles in AL. Während bei rein arithmetischen Rechnungen die AL offenbar im Vorteil ist, scheint die UPN bei stark verschachtelten Abläufen einen kleinen Vorteil zu besitzen. Die Klammern können durch die Anordnung der Stack-Speicher entfallen. Sie können nun eine kleine Ruhepause einlegen, um anschließend die obigen Beispiele nochmals durchzuarbeiten. Mit der Kenntnis der Funktionen der vier Stack-Speicher ist die größte Hürde im Bereich der UPN genommen. Wie wir schon gehört haben, ist es nahezu bei allen mit UPN ausgerüsteten PROTARen möglich, wenigstens in einer Richtung die Inhalte der vier Stack-Speicher zu vertauschen. Dies kann man sich zunutze machen, indem man einen Wert hin- und herwandern läßt, ohne ihn als Zwischenergebnis in einem Datenspeicher abstellen zu müssen. Wenn man den Weg dieses Wertes, den man dann später irgendwann einmal benötigt, genau verfolgt, genügt es, die Inhalte so oft umzuspeichern, bis die gewünschte Zahl in der Anzeige, dem X-Speicher, erscheint. Damit hat man einen unter Umständen nicht vorhandenen Datenspeicherplatz gewonnen. Doch von den Speichern soll erst später die Rede sein.

Erst bei der Programmierung kommen dann weitere Vorteile der UPN zum Tragen. Deshalb sollte man die einzelnen Systeme genau studieren, um für sich selbst eine ordentliche Entscheidungsgrundlage für die Rechnerauswahl zu schaffen.

2.2.3. Unterschiede

Eine Untersuchung zwischen den beiden Logikmethoden, der algebraischen Logik und der umgekehrten polnischen Notation hat den Vergleich als einen Wettkampf der ENTER-Taste (UPN) mit der Gleichheitstaste (AL) dargestellt. So einfach sollte man es sich nach meiner Meinung doch nicht machen. Jedes der beiden Systeme hat für bestimmte Verarbeitungen Vorteile, für andere Verarbeitungen Nachteile. Die beiden prominentesten Hersteller, die diese Logiken verwenden, sind in dem Fall als die eigentlichen „Wettkämpfer", anzusehen, die sich immer wieder etwas neues einfallen lassen und dadurch dem Anwender immer wieder neue Möglichkeiten eröffnen. Sicherlich wird man die eine oder andere Rechenmethode vorziehen, letzten Endes sollten Sie die Entscheidung nach den Funktionen des Rechners treffen, nämlich danach, ob er über diejenigen verfügt, die Sie zur Lösung Ihrer Probleme benötigen. Dies nur als kleine Einleitung zu einem Versuch, AL mit UPN zu vergleichen.

Das erste und äußerlich wesentlichste Merkmal sind die Gleichheitstaste (AL) und die ENTER-Taste (UPN). Mit letzterer ist selbstredend die Konstruktion des Stack, der vier Arbeitsspeicher, eng verbunden. Für AL ergibt sich daraus die Konsequenz, Klammertasten zu benötigen, um die Reihenfolge der mathematischen Operationen ändern zu können.

In einem abschließenden Kapitel soll versucht werden, unabhängig von verschiedenen Systemen, Hinweise zu geben, nach denen PROTARe einzelnen möglichen Verwendungen gemäß ausgewählt werden können. Kurz zusammengefaßt liegen die Hauptunterschiede zwischen AL und UPN in den erwähnten Tasten, den damit zusammenhängenden Klammertasten, dies rein äußerlich und innerlich im Unterschied im Aufbau der Arbeitsspeicher.

Wegen der Natürlichkeit der AL wird mit ihr vor allem in den Schulen gerne gearbeitet. Aber es sind deren mehrere Momente, die bei der Auswahl eines PROTARs eine Rolle spielen.

2.2.4. Das Zählen beginnt bei 0

Eine provokante Überschrift! Denn wir haben gelernt, mit dem Zählen bei der Zahl 1 zu beginnen und nun soll dies nicht mehr gelten? Wir müssen uns mit der Tatsache vertraut

machen, daß im Bereich der elektronischen Datenverarbeitung das Zählen tatsächlich bei der Zahl 0 beginnt, und das gilt gleichermaßen auch für uns, die wir mit PROTARen arbeiten wollen. Wir erkennen diesen Umstand bei der Lektüre der Bedienungsanleitung, die uns aufmerksam macht, daß der erste Speicher die Speicheradresse 0 trägt und die erste Programm-Marke ebenfalls eine Adresse 0 zur Kennzeichnung erhält. Wir wissen an dieser Stelle noch nichts über Speicher oder Programm-Marken, aber wir nehmen zur Kenntnis, daß die erste Position einer Reihe nicht die Zahl 1, sondern die Zahl 0 als Positionsnummer trägt. Aus der elektronischen Datenverarbeitung wissen wir, daß auch der Wert 0 ein elektrischer Zustand ist. Daß heißt, wenn ein Speicher den Wert 0 hat, ist sein Zustand „definiert". Ist der Speicher aber gelöscht, spricht der Fachmann von einem „nicht definierten" Zustand des Speichers.

Damit soll nur die Bedeutung der Zahl 0 hervorgehoben werden; denn wir müssen uns ja auch bei der Bedienung unseres PROTARs mit diesem Umstand anfreunden. Nehmen Sie beispielsweise die zehn einstelligen Zahlen, erkennen Sie sofort, daß diese Reihe auch die Zahl 0 umfassen muß, denn diese Reihe kann nur von 0 bis 9 laufen.

Ganz so kritisch ist es mit der Zahl 0 im Zusammenhang mit unseren PROTARen nicht. Hier trägt ein gelöschter Speicher den Wert 0, ohne daß wir uns über seinen Zustand Gedanken machen müssen. Aber die Zahl 0 als erste Adresse eines Speichers, bzw. einer Programm-Marke ist nicht wegzudiskutieren.

2.3. Tastenfeld

Das Tastenfeld ist gleichbedeutend mit der Eingabeeinheit eines Rechners. Über diese Eingabeeinheit gelangen sämtliche Anweisungen an den Rechner in sein geheimnisvolles Innere. Ein Unterschied zu Eingabeeinheiten großer Rechenanlagen besteht darin, daß bei PROTARen meines Wissens keine Texte eingegeben werden können; zumindest sind keine Alpha-Tastaturen vorhanden. Diese Einschränkung muß man machen, da heute ein Rechner (s. S. 120/6) existiert, dem mit Hilfe bestimmter Tastenfolgen bekannt gegeben werden kann, daß man sich einen Ausdruck bestimmter Buchstabenfolgen wünscht. Dies dient zur besseren Kennzeichnung von Ausgaben oder kann auch zur Bedienerführung, für einen Dialogverkehr zwischen Bedienung und Rechner, eingeplant werden. Mit Hilfe eines Dialogverkehrs kann man sich die Bearbeitung eines Problems erleichtern. Man setzt einen Text an bestimmte Stellen im Programm, der beispielsweise auf den nächsten, von der Bedienung auszuführenden Arbeitsgang, hinweist.

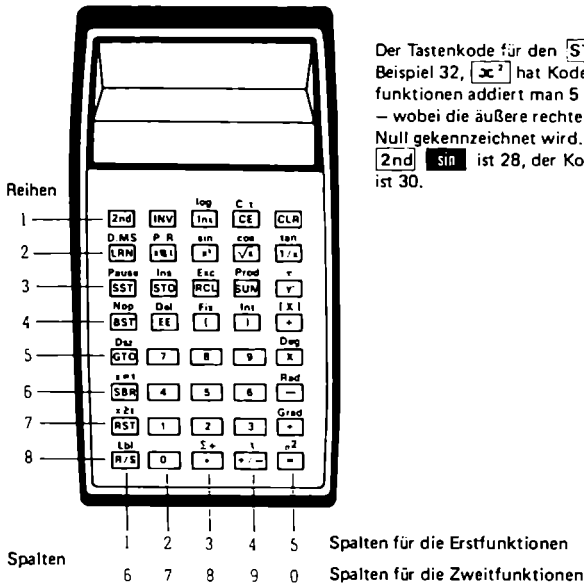
Neben dem Hauptschalter für Ein- und Ausschalten der Stromversorgung, können wir fünf Hauptgruppen von Tasten unterscheiden:

- Zahlentasten,
- mathematische Funktionstasten,
- Speicherbefehlstasten,
- spezielle Tasten zur Programmsteuerung und
- sonstige Tasten.

Der Unterschied zu den TARen besteht nun darin, daß bei PROTARen die zur Programmsteuerung notwendigen Tasten zusätzlich vorgesehen sind.

Wir wollen nun die einzelnen Bereiche des Tastenfeldes kennenlernen. Dabei ist es von untergeordneter Bedeutung, welches Gerät, bzw. welche Logik wir verwenden. Es soll versucht werden, eine umfassende Übersicht zu geben, sodaß man sich im Spezialfall nur mehr mit einzelnen Sonderfunktionen des Rechners, den wir gewählt haben, befassen muß. Es gilt auch hier, daß eine genaue Kenntnis der Vorgänge die Arbeit mit PROTAR wesentlich erleichtert, um im Zusammenhang mit der jeweiligen Bedienungsanleitung sämtliche gegebenen Möglichkeiten auch voll ausschöpfen zu können. Denn PROTAR ist

Beispiel eines Tastenfeldes eines mit der algebraischen Logik ausgestatteten Rechners (Texas Instruments TI 57).



Der Tastenkode für den STO-Befehl ist zum Beispiel 32, x^2 hat Kode 23. Für Zweitfunktionen addiert man 5 zur Spaltennummer – wobei die äußere rechte Spalte durch eine Null gekennzeichnet wird. Der Kode für 2nd sin ist 28, der Kode für 2nd π ist 30.

nicht mehr TAR, also ein Gerät, das ein wenig mehr als die vier Grundrechnungsarten beherrscht. So wollen wir uns ganz langsam und vorsichtig an die Geheimnisse der PROTARe heranmachen, die eigentlich gar keine sind, um mit ihnen vertraut zu werden.

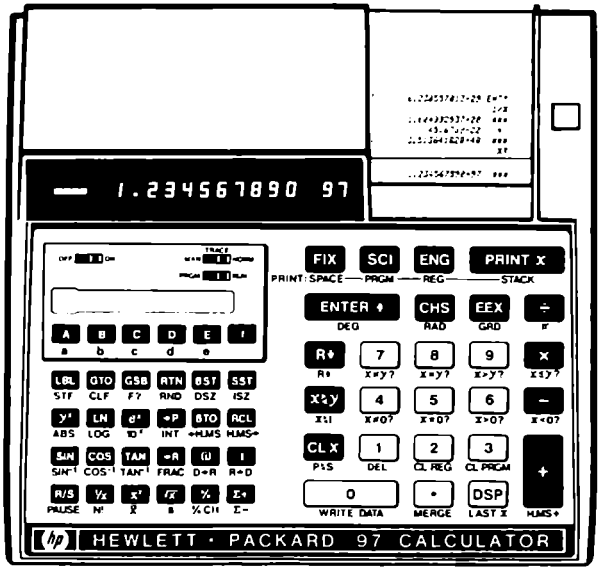
Bevor wir uns jedoch die einzelnen Bereiche des Tastenfeldes näher ansehen, nehmen wir einmal das gesamte Feld unter die Lupe. Dabei werden wir entdecken, daß neben der Hauptbezeichnung, die eine Taste direkt am Tastenkörper trägt, auch noch ein oder zwei weitere Bezeichnungen in der Nähe der Taste auf der Deckplatte angeordnet sind, meist in unterschiedlichen Farben gehalten. Die am Tastenkörper eingeprägte Funktion nennt man **Primärfunktion** (Erstfunktion), die anderen dagegen **Sekundärfunktionen** (Zweitfunktionen). Die Auslösung dieser Zweitfunktionen erfolgt über eine **Präfixtaste** (Vorwahltaste), die farblich mit dem Text der zugehörigen Zweitfunktion korrespondiert.

Bei der Besprechung einzelner Tasten werden die Vorwahltasten nicht erörtert, auch nicht dann, wenn sie von Fall zu Fall benötigt würden. Hier ist wieder auf die Bedienungsanleitung unseres PROTARs zu verweisen. Mit dem Vorhandensein einer oder mehrerer Vorwahltasten wird erreicht, daß trotz der Kleinheit des Tastenfeldes eine Vielzahl von Funktionen untergebracht werden kann. Die elektronische Einordnung ist leicht zu vollziehen. Schwierigkeiten bereitet unter Umständen die mechanische Auslösung, da für die Bedienung oft nur soviel Raum gelassen werden kann, daß beim Aufsetzen des Fingers leicht zwei Tasten zugleich berührt werden.

Beispiel eines Tastenfeldes eines mit der umgekehrten polnischen Notation ausgestatteten Rechners (Hewlett · Packard).

Automatischer Rechenregister-Stapel

T	0.000000000 00
Z	0.000000000 00
Y	0.000000000 00
Anzeige → X	1.234567890 91



Programmspeicher

001	51
002	51
003	51
004	51
005	51
006	51
007	51
008	51
009	51
010	51
219	51
220	51
221	51
222	51
223	51
224	51

Adressierbare Daten-Speicherregister

Primär-Speicherregister

I		25	R_9		9	R_{S9}		19	n
			R_8		8	R_{S8}		18	Σxy
R_E		24	R_7		7	R_{S7}		17	Σy^2
R_D		23	R_6		6	R_{S6}		16	Σy
R_C		22	R_5		5	R_{S5}		15	Σx^2
R_B		21	R_4		4	R_{S4}		14	Σx
R_A		20	R_3		3	R_{S3}		13	
			R_2		2	R_{S2}		12	
			R_1		1	R_{S1}		11	
			R_0		0	R_{S0}		10	

Geschützte Sekundär-Speicherregister

Wir müssen auch in diesem Zusammenhang zweckentsprechend vorgehen und jene Geräteype auswählen, die für den angestrebten Bedarf optimalen Komfort bietet. Rechnen wir verhältnismäßig wenig, könnten wir die Tasten auch mit feinen Stiften erreichen und damit die Funktion auslösen, d. h. ein sehr kleines Modell wählen. Bei länger andauernder Nutzung wird ein sehr kleines Tastenfeld oft zur Mühsal und sollte daher vermieden werden. Das gleiche gilt auch für die Anzeige der Ergebnisse. Haben wir ein Modell mit sehr kleinen Ziffern in der Anzeige, werden wir bei länger andauernder Arbeit rasch ermüden. Hier hilft entweder eine größere Anzeige oder ein mit einem Drucker ausgestattetes Modell. Obwohl im Druck die Zeichen nicht wesentlich größer sind als in der Anzeige, so scheint es mir, daß die Augen durch den Druck weniger belastet sind, als durch gleich große Zeichen in der Anzeige. Zur Entscheidung, welcher Rechner der für Sie der geeignetste wäre, können auch die Hinweise dienen, die bei den einzelnen Auswahlkriterien im Kapitel 7 angedeutet werden.

2.3.1. Das Zahlenfeld

Zum Zahlenfeld gehören die Ziffemtasten 0 bis 9 und der Dezimalpunkt. Die Anordnung ist genormt, wobei allerdings die Stellung der 0 und des Dezimalpunktes variieren kann.

7	8	9
4	5	6
1	2	3
0	.	

Die vorliegende Anordnung ist sehr häufig anzutreffen. Allerdings ist mir persönlich im Zusammenhang mit PROTARen nicht bekannt, ob es Modelle mit den Tasten 00 und 000 gibt. Solche Tasten sind manchmal bei kaufmännischen Rechengeräten anzutreffen. Es scheint hierfür bei PROTARen auch die Notwendigkeit dazu zu fehlen. Bei der Kleinheit des Tastenfeldes sind mehrere Nullen auch relativ rasch gesetzt und andererseits gibt es für Operationen mit den Zahlen 100, 1.000, 10.000, etc. zum Zweck der Dezimalpunktverschiebung auch andere, bessere Methoden, die im folgenden Abschnitt besprochen werden.

2.3.2. Mathematische Funktionen

Operationstasten der vier Grundrechnungsarten sind auf jedem Rechner vorhanden.

+ steht für die Addition;
− für die Subtraktion;
× für die Multiplikation und
÷ für die Division. Dieses Zeichen hat neben den beiden untereinander liegenden Punkten dazwischen meist einen kleinen Querstrich.

Winkelfunktionen, bzw. deren Umkehrfunktionen gehören. ebenso zur Standardausrüstung der Rechner;

sin entspricht der Sinusfunktion;
cos der Cosinusfunktion;
tan der Tangensfunktion.

Dem, der sich dabei an seinen Schulunterricht in Mathematik erinnert, wird das Fehlen der Cotangensfunktion auffallen. Er wird sich allerdings bei näherem Hinsehen vielleicht auch daran erinnern, daß diese Funktion nur den Kehrwert der Tangensfunktion darstellt und somit bei der Ausstattung der Rechner mit der Funktion $^1/_x$ verzichtet werden kann.

Nach Berechnung des Tangens eines Winkels wäre also die $^1/_x$ - Taste zu drücken, um den Cotangens zu erhalten.

LN bezeichnet den natürlichen Logarithmus,
LOG den dekadischen.
e^x So bezeichnet man die e-Potenz oder die natürliche Potenz und mit
10^x die dekadische. Mit dieser Taste wären nun die Stellenwertverschiebungen bequem und ohne viele Nullen drücken zu müssen, zu bewerkstelligen, indem man mit X die Anzahl der Dezimalstellen eingibt, um die man den Dezimalpunkt verschoben haben will. Dazu ein Beispiel:

Eine Multiplikation der Zahl 2,5 mit der Zahl 100 könnte wie folgt ausgeführt werden:

$$2{,}5 \times 10^2.$$

Wir hätten dazu nachstehend Tasten in AL in der angegebenen Reihenfolge zu betätigen, um diese Rechenaufgabe zu lösen:

2

5

x

2

10^x (oder auch INV LOG)

=

Damit haben wir den Dezimalpunkt um zwei Stellen nach rechts verschoben. Hier ist allerdings ein wenig Vorsicht am Platz, denn die Schaltung dieser Funktion erfolgt unter Umständen nach einem Prinzip, das hinter dem Dezimalpunkt nicht nur Nullen hinterlassen muß. Daher sollte diese Stellenwertverschiebung besser mit einer anderen, der

EE- oder EEX-Taste ausgeführt werden. Diese Tasten sind ebenfalls Exponentialtasten und werden im Falle der EE-Taste in der AL-Logik auch zur Anwahl der wissenschaftlichen Zahlendarstellung benutzt. Die Tastenfolgen 1 EE 2, bzw. EEX 2 bedeuten 10^2, also 100.

Beispiel:
Die gleiche Multiplikation, wiederum in der AL-Logik, sei nun in der anderen und sicheren Variante ausgeführt:

2

5

x

1

EE

2

=

Dann gibt es Tasten, die anzuwählen sind, wenn Sie Winkel anstelle in Altgraden – eine Darstellung, die standardmäßig nach dem Einschalten des Rechners erfolgt – in Neugraden oder im Bogenmaß berechnen wollen. Nicht zu vergessen sind die %-Taste, Quadrattaste (x^2), Wurzeltaste, Potenztaste (y^x) und die Pi-Taste.

Ein Wort noch zur Zahlendarstellung. In diesem Zusammenhang kann auch vom Begriff **Zahlenformat** gesprochen werden. Wir kennen die Begriffe **Fixkomma** und **Gleitkomma**. Von einem **Fixkomma** ist dann die Rede, wenn im Zuge der Ausgabe eines Ergebnisses dieses mit einer fix angewählten Anzahl an Dezimalstellen angezeigt oder ausgedruckt wird. Die Anzeige ist im allgemeinen kaufmännisch gerundet. Dies hat aber auf weitere Berechnungen mit dieser Zahl rechnerintern keinen Einfluß, da diese Zahl mit allen zur Verfügung stehenden und im Rechner darstellbaren Dezimalstellen weiter verarbeitet wird. Mit einem auf **Gleitkomma** eingestellten Rechner erhalten Sie jeweils soviele Dezimalstellen angezeigt, wie dies einerseits vom Ergebnis her bedingt ist, allerdings nur so viele, wie es die Rechnertype bzw. die Kapazität der Anzeige zuläßt. Für die Eingabe gilt in allen Fällen das Gleitkomma.

Dann gibt es noch die **technische,** bzw. **wissenschaftliche Zahlendarstellung,** Darüber kann der Interessierte in der jeweiligen Bedienungsanleitung nachlesen, da sie für unsere Diskussionen von untergeordneter Bedeutung ist. Die wissenschaftliche Darstellung wird vom Rechner selbständig aber dann gewählt, wenn das Ergebnis so klein oder so groß ist, daß es keine andere Wahl der Darstellung gibt, weil eben nur 8 oder 12 Stellen angezeigt werden können.

Nur zur Information sei die Zahl 225 in wissenschaftlicher Darstellung angeschrieben: 2,25 02; darin bedeutet die Zahl 02 den Exponenten zur Basis 10. Denn genau angeschrieben müßte die Darstellung so lauten: $2{,}25 \times 10^{02}$. Die Zahl 0.225 würde dagegen wie folgt aussehen: 2,25-01 in der Anzeige des Rechners oder komplett dargestellt $2{,}25 \times 10^{-01}$. Wir sprechen daher auch von einer Exponentialdarstellung. Diese Art von Zahlendarstellung kann ebenso wie die normale Darstellung mit einem Fixkomma versehen werden.

Dazu gibt es noch eine Reihe statistischer Funktionen:

$\Sigma+$ steht für die Aufrechnung zu Summen für die Bildung von Mittelwerten und Standardabweichung;

$\Sigma-$ damit wird die Summe und sämtliche anderen damit zusammenhängenden Daten korrigiert;

$\overline{x}$ ist der Mittelwert;

s die Standardabweichung;

$\hat{y}$ mit dieser Taste kann bei einigen Modellen eine lineare Regression ausgeführt werden.

N! Diese Funktion N!-Fakultäten kennen im allgemeinen auch nur mehr die Mathematiker; z. B. bedeutet 5! folgende Multiplikation 1.2.3.4.5. = 120.

Kompliziertere mathematische Funktionen finden wir im allgemeinen nicht mehr in fixen Schaltkreisen in den Rechnern eingebaut.

Zum Abschluß unserer Erläuterungen über das Zahlenfeld scheint es angebracht, auch ein paar Worte der **Zahleneingabe** zu widmen.

Der Rechner nimmt so lange eine Zahlenreihe als zusammengehörend an, bis entweder die Kapazität des Speichers überschritten ist, oder irgendeine andere Taste, als eine Ziffern- oder Dezimaltaste, gedrückt wird. Mit der Betätigung einer solchen Taste erkennt der Anzeigespeicher die **Beendigung der Zahleneingabe** an. Das ist sehr wesentlich zu wissen, um vor allem bei in Programmen gesetzten Zahlen den gleichen Effekt zu erzielen, wie dies uns – seien wir ehrlich – schon seit jeher auf einer Rechenmaschine als Selbstverständlichkeit schien.

Noch ein Wort zu den **Kettenrechnungen**. Für PROTARe ist sie wohl in allen Fällen kein Problem. Speziell in der algebraischen Rechenlogik wird ein komplexer Rechenvorgang erst mit der Betätigung des Gleichheitszeichens vollständig abgeschlossen. Wollen wir aber trotzdem weiterrechnen, ist zur Neueröffnung einer Kette sofort die erste Rechenoperation durch Betätigung einer der vier Funktionstasten anzuwählen. Bei UPN ist dies naturgemäß etwas anders, da ja die Funktion erst am Ende, d. h. nach Eingabe des zweiten Operanden ausgelöst wird.

2.3.3. Prozentrechnung

Wer häufig Prozentrechnungen durchzuführen hat, sollte einen PROTAR mit einer %-Taste wählen, wie sie im übrigen auch die meisten TARe aufweisen.

An sich ist eine Prozentrechnung nicht unbedingt an das Vorhandensein einer derartigen Taste gebunden. Wie wir in einem späteren Beispiel noch erläutern werden, kann eine Aufrechnung eines Prozentsatzes auf einen Grundwert rechentechnisch auch in anderer Form ablaufen, und zwar:

$$P_n = P_a \times (1 + p/100).$$

P_n sei der neue Preis, mit P_a der alte bezeichnet und p der Prozentsatz der Aufrechnung.

Nun gibt es aber Fälle, in denen es wichtig ist, den Betrag der Aufrechnung zu kennen und nicht nur den neuen Wert. Das erfordert ohne Prozenttaste etwas mehr Aufwand, nämlich:

- Eingabe des Grundwertes und Abspeicherung.
- Da sich der Grundwert noch in der Anzeige befindet, kann der Betrag der Aufrechnung ermittelt werden.
- Eingabe des Prozentsatzes.
- Division durch 100.
- Multiplikation mit dem Grundwert; damit wird der Betrag der Aufrechnung angezeigt.
- Abspeicherung, d. h. Addition dieses Betrages zum abgespeicherten Grundwert und
- Abruf des Gesamtbetrages in das Anzeige-Register.

Unter Verwendung einer %-Taste liest sich der gleiche Vorgang wie folgt:

- Eingabe des Grundwertes.
- Eingabe des Prozentsatzes.
- Drücken der Prozenttaste, wobei der Betrag der Aufrechnung angezeigt wird und
- Aufsummierung zum Gesamtbetrag.

So wird im allgemeinen auch ein Aufschlag mit UPN berechnet, wie z. B. bei der Verrechnung der Mehrwertsteuer. In diesem Fall befindet sich der Grundwert im zweiten Arbeitsspeicher, sodaß sofort die Gesamtsumme gebildet werden kann.

2.3.4. Speicherbefehle

Von den einfachen Taschenrechnermodellen wissen wir, daß diese, wenn überhaupt, meist nur einen Speicher haben, in den ein Zwischenergebnis abgestellt werden kann, mehr ist in den seltensten Fällen vorhanden. Unser PROTAR dagegen ist im Besitze von mehr als einem Speicher, in die im Verlaufe der Rechenoperationen Werte abgelegt werden, um sie zu einem späteren Zeitpunkt wieder abrufen und weiterverarbeiten zu können. Wir werden später über die Speicher noch ausführlich sprechen müssen, hier sei aber nur so viel gesagt, als zum Verständnis nötig ist, denn wir wollen hier ja wissen, wie die Zahlen in den Speicher hineinkommen, und wie man sie wieder von dort herausbekommt.

Die im folgenden dargestellten Speicherbefehle finden wir unter Umständen in der gleichen Form auch bei den TARen. Sie haben auch selbstredend die gleiche Funktion. Dort genügt es aber im allgemeinen, diese Speichertasten allein zu betätigen, weil man ja nur einen Speicher hat – es wäre auch wenig sinnvoll, TARe mit mehr als einem Speicher auszustatten. Nachdem PROTARe, wie schon erwähnt, mehr als einen Speicher aufweisen, muß nach dem Drücken der Speichertasten zum Abspeichern und zum Abrufen auch eine **Adresstaste** gedrückt werden. Da bei mehreren Speichern diese voneinander zu unterscheiden sind, erhalten sie meist fortlaufende Nummern von 0 beginnend als Adresse.

STO Ist die Taste, mit der ein Wert aus der Anzeige in einen Speicher abgestellt wird. Danach muß eine Zifferntaste gedrückt werden, um den Speichervorgang endgültig

auslösen zu können. STO bedeutet „store" und heißt nichts anderes als „abspeichern", „lagern".

RCL ist die Taste, mit der ein Wert aus einem Speicher wieder in die Anzeige zurückgerufen werden kann. Auch hier muß nach dem Drücken dieser Taste eine Zifferntaste folgen, um den Vorgang komplett zu machen. RCL heißt „recall" und bedeutet nichts anderes als „wieder rufen" oder „zurückrufen".

C, CE, CLX, CLR sind verschiedene Formen von Löschtasten, wobei die ersten drei jeweils nur das Eingaberegister, d. h. den Eingabespeicher, der vierte auch die Datenspeicher löscht. Gerade auf dem Sektor der Löschtasten herrscht meines Erachtens ein heilloses Durcheinander, da fallweise die unterschiedlichsten Löschfunktionen miteinander gekuppelt sein können. Daher kann hier keine allgemein gültige Regel aufgestellt werden. Am besten Sie informieren sich in der jeweiligen Bedienungsanleitung. Eines haben alle Löschtasten jedoch gemeinsam: den Anfangsbuchstaben C. Denn C ist auch der Anfangsbuchstabe von „clear", was soviel heißt wie „löschen".

Gestatten Sie, sehr verehrter Leser, noch einige Worte zum Löschen von Speichern.

Das Löschen betrifft - wenn Sie die entsprechende Taste CLR betätigen - im allgemeinen sämtliche Speicher auf einmal. Sie wollen aber fallweise Speicher löschen, ohne dabei andere zu berühren. In diesem Fall ist wie folgt vorzugehen:

1. 0 eintasten, d. h. in das Anzeigeregister bringen;
2. STO-Taste drücken;
3. Zifferntaste als Adresstaste drücken.

Damit haben wir auf einfache Weise den betreffenden Speicher gelöscht ohne die übrigen Speicherinhalte verändert zu haben. Dieser Vorgang ist während eines Programms oft nötig, wenn sich immer wieder wiederholende Vorgänge – z. B. Kalkulationen, wo zum Beispiel der Speicher, der zum Aufsummieren der Kosten verwendet wird, vor jeder neuen Rechnung den Wert 0 annehmen muß - ergeben.

Im Rahmen der Speichertasten haben wir es auch mit einer besonderen Art von Speichertasten zu tun, den Speichertauschtasten. Diese werden wohl vorzugsweise zur Steuerung von Programmabläufen verwendet und verursachen einen Tausch des Speicherinhaltes des Anzeigeregisters mit einem noch näher bezeichnenden Datenspeicher.

x ≷ y zeigt den Speichertausch zwischen der Anzeige, also dem ersten, und dem zweiten Arbeitsspeicher; dies ist nicht nur bei UPN, sondern auch fallweise bei AL möglich.

Exc gefolgt von einer Zifferntaste ermöglicht den Tausch des Inhalts der Anzeige mit dem durch die Zifferntaste angesprochenen Datenspeicher.

x ≷ t ermöglicht den Tausch der Speicherinhalte der Anzeige und des Testregisters; dieses wird bei Abfragen benötigt, doch davon später mehr.

x ≷ I bewirkt den Tausch der Speicherinhalte des Anzeige- mit dem Indexregister.

Auf eine Besonderheit – die leider nur bei Rechnern eines Herstellers (s. S. 120/1) vorgesehen wird – soll hier eingegangen werden, weil sie einen ganz besonderen Speicher betrifft. Es handelt sich um den „Last X"-Speicher. In diesen Speicher wird bei Veränderung der Anzeige der jeweils letzte Wert des Anzeigeregisters abgestellt und damit der vorletzte Wert überschrieben. Für diese Art der Speicherung ist kein besonderer Vorgang nötig; einzig beim Abruf ist die „Last X"-Taste zu betätigen und der alte Wert des Anzeigeregisters wird in die Anzeige zurückgeholt. Diese Einrichtung ist nicht allein im Falle der Verarbeitung von Programmen, sondern auch bei manueller Verarbeitung von Bedeutung. Anläßlich einer jüngst veranstalteten Büromaschinenfachmesse wurde von

einem Hersteller ein TAR vorgestellt, der die jeweils letzten 20 Eingaben speichern kann, Wie weit diese Eigenschaft in der Praxis nutzbringend angewendet werden kann, bleibt abzuwarten. Ebenso ist auch zu überlegen, ob damit auch PROTARe ausgestattet werden.

2.3.5. Programmbefehle

Mit diesem Kapitel kommen wir zum ersten Mal im Verlaufe unserer Ausführungen unseren eigentlichen Absichten näher, nämlich dem PROTAR ein wenig auf die „Finger", sprich Tasten, zu sehen und zu beobachten, wie er denn das macht mit dem „Abarbeiten" eines Programmes. Unter diesem Begriff des „Abarbeiten" versteht der Eingeweihte die Programmausführung durch den Rechner.

Die hier im einzelnen zu diskutierenden Tasten werden nur im Zusammenhang mit Programmen benötigt und dienen dazu, die einzelnen Programmzweige voneinander zu trennen, bzw. den Ablauf eines Programmes zu steuern.

R/S Start/Stop-Taste oder „run/stop"; mit dieser Taste allein kann schon ein Programm „gefahren" werden, indem man es durch Drücken dieser Taste startet. Das Ende findet das Programm dort, wo wir die gleiche Taste im Programmablauf vorgesehen haben, und hält an, sodaß wir das Ergebnis ablesen können.

SST bedeutet „single step" und ermöglicht ein Programm schrittweise abzuarbeiten. Im Programmiermodus – d. h. bei der Schalterstellung, in der ein Programm eingegeben wird - ist die Funktion ähnlich, sie stellt den Programmzeiger auf den nächsten Programmschritt ein, wobei Zeilennummer und Tastencode angezeigt wird; in diesem Fall erfolgt keine Programmausführung.

BST heißt „back step". Diese Taste bewirkt genau das Gegenteil, was unter SST im Programmiermodus dargestellt wurde.

RTN „return" beendet einen bestimmten Programmzweig in UPN; der Rechner hält an dieser Stelle an. Ausnahme: wenn RTN das Ende eines Unterprogrammes (siehe bei GSB oder SBR, bzw. Abschnitt 3.5.) anzeigt. Die gleiche Wirkung hat die Tastenfolge INV SBR, die in AL verwendet wird. Im Rechenmodus – Schalterstellung des Rechners auf Rechenbetrieb eingestellt – bewirkt die Betätigung der RTN – Taste die Einstellung des Programmzeigers auf den Programmanfang.

RST „reset" stellt ebenfalls den Programmzeiger in AL auf den Programmanfang ein, und zwar ebenfalls im Rechenmodus.

LBL „Label" oder Marke fixiert Im Zusammenhang mit einer anderen Taste der Label-Adreßtaste, eine bestimmte Stelle im Programm, die Sie direkt ansteuern können, wenn es sich um einen vom Programm aus bestimmten Label handelt. Einen Label oder eine Marke benützt man auch, um ein Programm starten zu können.

GTO „go to" ermöglicht einen unbedingten Sprung zu einer bestimmten Marke, d. h. diese Taste kann ebenfalls nicht allein gedrückt werden, sondern wirkt nur im Zusammenhang mit einer zweiten Taste, der Labeladresse. Diese Tastenfolge veranlaßt das Programm – wenn es diesen Befehl, diese Anweisung erreicht – den normalen Ablauf zu verlassen und die Verarbeitung an der so bezeichneten Stelle fortzusetzen.

GSB oder SBR „go to subroutine" ist eine Anweisung, die Verarbeitung beim bezeichneten Label fortzusetzen und nach Erreichung der Anweisung RTN oder INV SBR zum nächsten Befehl ins Hauptprogramm zurückzukehren. Man spricht hier von einem Sprung in eln Unterprogramm mit automatischer Rückkehr ins Hauptprogramm. GSB und RTN gilt im Fall UPN, SBR und INV SBR im Fall AL. Will man aber unter Umständen kelne Rückkehr ins Hauptprogramm, wäre an die Stelle von RTN, bzw. INV SBR der R/S-Befehl zu setzen. Es braucht nicht weiter betont zu

werden, daß, gleich wie beim Sprungbefehl, eine zweite Taste mit der Labeladresse zu drücken ist, damit die Anweisung vollständig ist und ausgeführt werden kann.

Nun kommen wir zu den wichtigsten Tasten innerhalb dieses Kapitels. Sie sind sozusagen das Salz in der Suppe des Programmierers. Mit ihnen wird ein Programm belebt, sodaß verschiedene Möglichkeiten, ein Programm ablaufen zu lassen, eröffnet werden.

Die Tasten, die **Vergleichstasten**, stellen eine **Bedingung** auf, nach der im Falle der Antwort ja oder nein das Programm durch einen Sprung verzweigt werden kann. Wir sprechen dann von einem **bedingten Sprung**. Dies nur als Hinweis; mehr darüber im Kapitel der Elemente der Programmierung. Grundsätzlich wird im Falle der Antwort ja, d. h. im Falle, daß die gestellte Bedingung erfüllt ist, der nächste Befehl, der je nach Modell eine Sprunganweisung sein kann oder muß, (bitte hier die Bedienungsanleitung genau studieren!) ausgeführt. Im Falle nein wird der nächste Befehl, der ein Sprungbefehl sein kann, ignoriert und die Verarbeitung beim übernächsten Befehl fortgesetzt. Soviel hier zum besseren Verständnis der Vergleichsbefehle.

$x = t$ Der Inhalt der Anzeige wird mit dem Inhalt des Testregisters verglichen. Der nächste Befehl wird ausgeführt, wenn beide Speicherinhalte gleich sind.

$x \geqslant t$ Der nächste Befehl wird ausgeführt, wenn der Wert in der Anzeige größer oder gleich dem Inhalt des Testspeichers ist.

$x = y$ Der Inhalt von zwei Arbeitsspeichern wird miteinander verglichen; der folgende Befehl wird nur ausgeführt, wenn beide Inhalte einander gleich sind und das bis auf die letzte Stelle, die der Rechner intern gespeichert hat! Diese Bemerkung ist sehr wichtig, da eine Gleichheit im folgenden Zahlenbeispiel nicht gegeben ist. $x = 1{,}573865431$; $y = 1{,}573865430$. Die gestellte Bedingung $x = y$ ist also hier wohl größenordnungsmäßig, nicht aber exakt mathematisch erfüllt.

$x = y$ Der nächstfolgende Befehl wird im Gegensatz zum vorigen Fall nur bei Ungleichheit beider Werte ausgeführt.

$x \leqslant y$ Das gleiche gilt für den Fall, daß der Wert in x kleiner oder gleich dem Wert in y ist.

$x > y$ Nur für den Fall, daß der in x gespeicherte Wert größer ist, als der in y, wird der nächste Befehl ausgeführt.

Dann gibt es auch eine meines Wissens nur von einem Hersteller (s. S. 120) benützte Reihe von 0-Abfragen.

$x = 0$ Der nächste Befehl wird dann ausgeführt, wenn der Wert in der Anzeige 0 ist.

$x \neq 0$ Das gleiche gilt für den Fall, daß der Wert der Anzeige ungleich 0 ist.

$x < 0$ Die Ausführung des nachfolgenden Befehls erfolgt nur in dem Fall, in dem der Wert der Anzeige kleiner als 0 ist.

$x > 0$ Analog dazu gilt, wenn der Wert der Anzeige größer als 0 ist.

In einigen Fällen finden wir bei diesen Funktionen auch ein Fragezeichen, denn ein Vergleich ist ja mit einer Abfrage gekoppelt.

Ohne unseren weiteren Ausführungen vorgreifen zu wollen, kann gesagt werden, daß die mit den Abfragen gekoppelten Programmverzweigungen zweifellos das Herzstück von Programmen sind. Sie ermöglichen in Abhängigkeit von gestellten Bedingungen unterschiedliche Lösungsmöglichkeiten in einem Programm unterzubringen. Mehr davon allerdings erst später in den entsprechenden Abschnitten.

Zur Programmkorrektur – wer wird diese Fälle nicht benötigen? – brauchen wird aber noch einige Tasten, und zwar

DEL „delete", womit ein Programmschritt gelöscht wird, und

INS „insert", mit Hilfe der ein zusätzlicher Befehl eingeschoben werden kann.

Auch hier gibt es zwischen den Rechnermodellen Unterschiede, und zwar ist in einem Fall kein Befehl INS nötig, da bei Stellung des Programmzeigers auf eine bestimmte Zeile jeder im Programmiermodus eingetastete Befehl zwischen dem angezeigten und den übrigen Befehlen automatisch eingeschoben werden kann. Die nachfolgenden Befehle werden dabei um eine Programmzeile verschoben. Auch hier ist wieder, der Sicherheit halber, auf die jeweilige Bedienungsanleitung zu verweisen. Bei jenen Rechnern, wo INS verlangt wird, würde – ginge man wie zuvor beschrieben vor – der nachfolgende Befehl ohne Drücken der INS-Taste überschrieben werden. In diesem Fall bewirkt das Drücken der INS-Taste das Verschieben der nachfolgenden Befehle um je eine Programmposition.

2.3.6. Sonstige Tasten

Zum Abschluß über die Diskussion des Tastenfeldes wollen wir uns noch einige Sondertasten ansehen, die in unterschiedlicher Anzahl vorhanden sein können.

Sicherlich vorhanden sind die **Präfix-Tasten,** die den Rechner anweisen, bei Drücken einer Taste nach Betätigung der Vorwahltaste, wie diese Taste auch genannt wird, die **Sekundärfunktion** zu wählen. Bezeichnet sind solche Tasten beispielsweise mit 2nd, f, g und h. Sie sind meist auch unterschiedlich gefärbt.

Bei den Rechnern eines Herstellers (s. S. 120/2) gibt es eine weitere Besonderheit, die INV-Taste. INV heißt nichts anderes als „inverse" und bedeutet Umkehrung. Die nach Betätigung dieser Taste angewählte Funktion wird umgekehrt. Mit der INT-Taste wird beispielsweise die Berechnung des ganzzahligen Teils einer Zahl ausgelöst. INV INT bewirkt das Gegenteil, also die Berechnung des Dezimalteiles. INV SUM bedeutet das Gegenteil einer Aufsummierung, nämlich den Abzug der angezeigten Zahl. Mit INV LOG wird nicht der dekadische Logarithmus einer Zahl berechnet, sondern die Umkehrung, nämlich 10^x. Mit dieser ausgezeichneten Lösung ist es dem Hersteller der mit der INV-Taste ausgestatteten Rechnern gelungen, noch mehr Funktionen auf einem an sich kleinen Tastenfeld unterzubringen.

Dann gibt es Tasten, die in der Lage sind, in Stunden, Minuten und Sekunden angegebene Zeitwerte in Dezimalstunden umzuwandeln, bzw. den umgekehrten Vorgang auszulösen. Bei einigen Modellen gibt es die Möglichkeit, für allfällig später einmal nachträglich im Programm einzufügende Anweisungen den hiefür nötigen Programmspeicherplatz schon bei der ersten Programmierung mit der Taste NOP – sie bedeutet „no operation", ist also eine sogenannte Leeranweisung, die im Programmablauf ignoriert wird – zu reservieren.

Fast alle Rechner kennen die DSZ-Anweisung, mit der eine Programmschleife gebaut werden kann. Als Schleife bezeichnet man eine Programmfolge, die so lange und immer wieder durchlaufen wird, bis sie aufgrund einer gestellten Bedingung verlassen werden kann. Erst dann kann die Verarbeitung normal forgesetzt werden. Mehr erfahren Sie darüber im Abschnitt 3.4.3. Diese Anweisung bewirkt eine sukzessive Reduzierung eines Zählers – nach jedem Durchlauf der Schleife wird dieser Zähler mit der DSZ-Anweisung um den Wert 1 vermindert – bis das Programm bei Erreichen des Zählerinhalts 0 die Schleife verläßt und normal fortsetzt.

Für die Indexspeicher zum indizierten Rechnen – siehe Abschnitt 2.6.2. – gibt es verschiedene Formen. Es soll in diesem Zusammenhang nur auf die Existenz solcher Möglichkeiten hingewiesen werden; in den Bedienungsanleitungen sollte Näheres darüber nachgelesen werden.

Abschließend soll noch auf eine Taste besonders eingegangen werden, die in der letzten Zeit meines Wissens bereits in fast alle Rechnertypen eingebaut wurde, und zwar ist das die PAUSE-Taste. Sie hat natürlich nur ihre Bedeutung bei Abarbeitung von Programmen, wo es darum geht, Zwischenergebnisse kurzzeitig – etwa 1/2 bis 1 Sekunde lang – anzuzeigen und dann das Programm automatisch fortzusetzen. Würde das Programm

aufgrund der vorgesehenen R/S-Anweisung anhalten, müßten Sie es mit der R/S-Taste erneut in Gang setzen, was bei dieser Taste entfällt. Längere Anzeigedauer erreicht man durch mehrfache Betätigung der PAUSE-Taste.

Falls Sie einen PROTAR mit Druckeinrichtung zu betreiben beabsichtigen, dann sei noch kurz auf Sondertasten hingewiesen, die es ermöglichen, vom Programm aus einen Papiervorschub um eine Zeile und den Andruck des augenblicklich in der Anzeige befindlichen Speicherinhalts zu bewirken. Denn nichts stört so sehr die Leserlichkeit von reinen Zahlenergebnissen, wenn man zwischen einzelnen Zahlengruppen nicht Abstände einschiebt. Es nützt allein wenig, wenn sie beispielsweise Ergebniszahlen mit einem Index, den Sie ohne Dezimalstellen ausdrucken, einleiten wollen. Diese Zahlengruppe sollte sich von der vorhergehenden deutlich abheben.

Der Vollständigkeit halber sei auch auf Tasten hingewiesen, mit denen man das Auslisten der Programmbefehle, sämtlicher Label, der Speicherinhalte, etc. auslösen kann. Diese Tasten kommen in erster Linie bei Tests und Kontrollen der Programme zum Tragen.

2.4. Ausgabe

Was das Tastenfeld für den Eingabebereich darstellt, ist die Anzeige oder der Druck für den Ausgabesektor. Hier ist nur zum **Ausgabeformat** etwas zu sagen.

Unter dem Begriff „Ausgabeformat" versteht man die Art, bzw. die Form einer Zahl, in der diese ausgegeben wird. Sie kann in normaler oder wissenschaftlicher Darstellung angezeigt oder ausgedruckt werden. Im Begriff „Ausgabeformat" ist auch die Anzahl der Dezimalstellen einer Zahl enthalten. Im übrigen wird hier nochmals auf den Abschnitt 2.3.2. verwiesen.
Wie schon im Abschnitt 2.3.2. angedeutet, gibt es für die Ausgabe, d.h. für die Darstellung der Ergebnisse von der Anzahl der Dezimalstellen her zwei Möglichkeiten. Die erste ist das **Fixkomma**, d. h. das Ergebnis wird mit der vom Rechner selbst oder vom Benutzer fix eingestellter Anzahl an Dezimalstellen ausgegeben. Manche Rechner sind beim Einschalten automatisch auf zwei Dezimalstellen eingestellt. Zum zweiten haben wir das **Gleitkomma**, d. h. ein Ergebnis wird mit jener Anzahl an Dezimalstellen ausgegeben, als es die Rechnung verlangt. Zum Beispiel wird das Ergebnis unseres ersten Rechenbeispieles

$$12 + 9 = 21$$

mit 21. angedruckt, also mit 0 Dezimalstellen, das Ergebnis aus

$$2{,}3 \times 5 = 11.5$$

mit 11,5 also mit einer Dezimalstelle.

Nun gibt es hier allerdings eine Grenze, und zwar ist dies die Kapazität der Anzeige, d. h. die Anzahl an Ziffern, die ausgegeben werden kann. Diese Zahl bewegt sich zwischen 8 und 12.

Werden die Ergebnisse größer oder kleiner, als es die Kapazität der Anzeige zuläßt, schaltet der Rechner automatisch auf das Anzeigeformat in der Exponentialdarstellung. Entspricht die nächste Anzeige wieder der Anzeigekapazität, wird im allgemeinen wieder auf die Anzeige in der normalen Zahlendarstellung automatisch zurückgeschaltet. Diese Exponentialdarstellung kann aber auch angewählt werden. Die Rückführung in die normale Darstellung erfolgt dann nicht automatisch, sondern muß ebenfalls angewählt werden.

Rechnerintern wird meist mit mehr als den 8 oder 12 Stellen gearbeitet. Ausgegeben aber können diese zusätzlichen Stellen nicht werden.

Nachdem wir aber unser Schwergewicht auf die Erörterung kaufmännisch-betriebswirtschaftlicher Probleme legen, brauchen wir diese Exponentialdarstellung an sich nicht.

Trotzdem müssen wir sie kennen, um nicht von der Ausgabe eines exponentiell dargestellten Ergebnisses überrascht zu werden. Hier stellt sich die berechtigte Frage, ob wir nicht einen Fehler gemacht haben. Denn wir werden wohl kaum so große oder so kleine Ergebniszahlen in unserem Bereich vorfinden, daß wir diese Art von Zahlendarstellung einkalkulieren müssen. Wir überprüfen also unsere Rechnung sicherheitshalber.

2.4.1. Anzeige

Wenn Sie PROTAR konventionell benutzen, also ein Rechenproblem, ohne über ein Programm zu gehen, lösen, dann genügt es im allgemeinen, sich das Ergebnis über die Anzeige allein ausgeben lassen. Wenn Sie im Falle einer erforderlichen Dokumentation das Ergebnis allerdings gedruckt brauchen, müssen Sie den Druck über die entsprechende Taste PRINT oder PRINT X gesondert auslösen. Rechnet man außer mit Programmen auch konventionell und will man sich deswegen keinen zweiten Rechner zulegen, dann sollte man zu einem Rechner greifen, der sehr große Zahlen in der Anzeige hat. Sie schonen damit Ihre Augen, vor allem dann, wenn Sie über einen längeren Zeitraum ohne wesentliche Unterbrechungen arbeiten.

Wann braucht man die Anzeige in Verbindung mit Abarbeitung bei Programmen? Eine an sich berechtigte Frage. Denn man könnte sich ja damit begnügen, wesentliche Zwischenergebnisse und das Endresultat ausgedruckt zu erhalten.

Ein Fall ist der, daß Sie das Programm veranlaßt haben, an einer bestimmten Stelle unter Anzeige einer bestimmten Zahl im Display – ich möchte zur Gewöhnung hier einmal die englische Bezeichnung für die Anzeige wählen – anzuhalten, weil Sie hier eine Eingabe vorgesehen haben. Da genügt es vollauf, sich diesen Hinweis über die Anzeige geben zu lassen. Diese anzuzeigede Zahl muß nicht unbedingt ein Zwischenergebnis sein, es kann irgend ein Index als **Bedienerführung** gemeint sein. In diesem Fall haben Sie an dieser Programmstelle das Programm durch den Einbau der R/S-Anweisung halten lassen; daher müssen Sie das Programm auch wieder mit der R/S-Taste starten. Sie könnten das Programm ohne weiteres aber auch so abgefaßt haben, daß Sie bei gleicher Anzeige die nächste Eingabe über einen frei wählbaren Label vornehmen müssen.

Eine Anzeigevorrichtung ist in jedem Rechner vorhanden, auch wenn er mit einer Druckeinrichtung ausgestattet sein sollte. Handelt es sich bei dieser Art von Ausgabe um die Anzeige eines Zwischenergebnisses, dann können Sie sich dieses auch während einer Programmpause anzeigen lassen. Die PAUSE-Taste hat allerdings nur im Zusammenhang mit Programmen eine Bedeutung. Wenn Sie zum Beispiel einen Rechner in der Hand haben, von dem Sie glauben, es handle sich um einen TAR und Sie entdecken bei näherem Hinsehen auf oder neben einer Taste die Bezeichnung PAUSE, können Sie mit Sicherheit annehmen, einem Irrtum unterlegen zu sein. Denn PAUSE-Tasten gibt es nur auf PROTARen!

Während eines Programmablaufs können Sie mit dieser den Rechner veranlassen, nur kurz zur Anzeige eines Zwischenergebnisses anzuhalten, um hierauf von selbst wieder weiterzuarbeiten. Sie können die Pause aber verlängern, wenn Sie die PAUSE-Taste im Programm mehrmals hintereinander vorsehen. Dies wäre beispielsweise denkbar, wenn Sie sich ein Zwischenergebnis notieren müßten. In diesem Fall würde ich es mir aber lieber ausdrucken lassen.

2.4.2. Drucker

Druckeinrichtungen sind nur bei einer kleinen Zahl von Modellen vorhanden. Bei der Kleinheit dieser Geräte gibt es keine Farbbänder mehr, denn diese Einrichtungen arbeiten als sogenannte **Thermodrucker** nahezu wartungsfrei. Ein besonderes Papier ist allerdings vonnöten. In dieses wird das Druckbild eingebrannt. Dabei ist es ohne Bedeutung, ob der

Drucker im Rechner selbst eingebaut ist, oder ob der Rechner in irgendwelcher Form an einen Drucker angeschlossen wird. Über die Möglichkeiten, was gedruckt wird, haben wir uns eigentlich schon im letzten Abschnitt unterhalten. Bei diesen Modellen kann nicht nur gedruckt werden, was Sie im Verlaufe eines Programms vorsehen, sondern die Rechner sind auch in der Lage, die Programme vollständig aufzulisten, d h. sämtliche gesetzten Befehle Zeile für Zeile auszudrucken. Sie können sich auch den Inhalt sämtlicher Datenregister ausdrucken lassen, bei mit UPN arbeitenden Geräten kann auch der Inhalt der vier Arbeitsspeicher, des Stacks, ausgegeben werden. Ein Modell (s. S. 120/6) ist auch in der Lage, sämtliche Marken mit Angabe der Lage im Programm auszudrucken.

Vor allem die Möglichkeit der Programmauslistung unter Angabe von Zeilennummer, Befehlskennzeichen und des Tastencodes ist für die Redigierung und die Korrektur von großer Bedeutung. Suchen Sie einen Fehler, werden Sie mitunter erst einmal den Befehlsablauf kontrollieren und überprüfen, ob Sie sämtliche Befehle auch in der richtigen Reihenfolge Ihrem PROTAR bekanntgegeben haben. Aber wir wollen nicht weiter vorgreifen!

Wenn Sie den Druckstreifen als Dokumentation Ihrer Berechnungen als Beleg irgendwo einkleben wollen, müssen Sie vorerst probieren, ob sich der Kleber überhaupt dafür eignet. Lösungsmittelhaltige Klebstoffe tun dies meist nicht und verfärben das Papier, sodaß der Druck bis zur Unkenntlichkeit verschwindet. Für meinen Bedarf klebe ich die Streifen mit einem durchsichtigen Klebestreifen ein.

2.5. Speicher

Ein Speicher, auch Register benannt, ist eine bestimmte Stelle im Rechner, an der eine Zahl, ein Zwischenergebnis oder sonst irgendein Wert für eine spätere Verwendung abgestellt und damit archiviert werden kann. Ein Speicher wird aber auch benötigt, um einen Programmschritt aufzunehmen, damit der Programmzeiger immer weiß, wo im Programm er sich gerade befindet, bzw. welche „Arbeit" bzw. welche Anweisung als nächste auszuführen ist. Man könnte auch versucht sein, diesen Speicher als eine Art Programmkartei zu bezeichnen. Bei PROTARen ist der Speicher, der frei zugänglich ist, also zweigeteilt und dient als

- **Programmspeicher,** um sämtliche Anweisungen aufnehmen zu können und als
- **Datenspeicher;** hier ist nur kurz auf die im Abschnitt 2.3.4. über die Speichertasten gemachten Bemerkungen zu verweisen. Die Speicherung erfolgt bekanntlich mit der STO-Taste, der Abruf in das Anzeigenregister mit der RCL-Taste.

Es sei hier speziell auf eine Rechnertype (s. S. 120/7) hingewiesen, bei der der Benützer in der Lage ist, den Gesamtspeicher nach seinem Gutdünken zwischen den beiden Speicherbereichen aufzuteilen! Die maximal mögliche Datenspeicherzahl liegt hier bei 100, und die der Zahl an möglichen Anweisungen bei 960. Über die einzelnen Kapazitäten geben die Bedienungsanleitungen hinreichend Auskunft.

2.5.1. Programmspeicher

Ein Programmspeicher kann nur dann angesprochen werden, wenn der Wahlschalter auf Programmierung steht. Nur in diesem Modus ist es möglich, die einzelnen Programmschritte einzugeben.

Jeder Programmschritt ist gekennzeichnet durch

- die Zeilennummer;
- bis zu im allgemeinen drei Tastencodes.

Allein aufgrund dieser Code-Nummern läßt sich nachträglich feststellen, welche Taste(n) wir an welcher Stelle gedrückt haben und können dadurch überprüfen, ob wir unser Programm auch richtig eingegeben haben.

Dieser Speicher, daran sei hier erinnert, kann auch mit den Tasten SST nach vorne und BST nach rückwärts Schritt für Schritt kontrolliert werden. Dies ist durch die Betätigung der GTO-Taste und einer Zeilennummer im Rechenmodus möglich, um nicht den ganzen Programmspeicherbereich absuchen zu müssen. Man stellt den Programmzeiger damit auf eine ganz bestimmte Programmzeile. Der Programmzeiger wird an sich bei jedem kompletten Tastenbefehl automatisch um einen Schritt weiter gestellt. Bezüglich der Korrekturen sei an die Bemerkungen im Programm-Modus erinnert, die wir anläßlich der Besprechung der Tasten DEL für das Entfernen eines Befehls, und INS für das Einfügen eines Befehls kennengelernt haben und weisen nochmals darauf hin, daß hier in der Arbeitsweise Unterschiede bestehen, über die man sich am besten in der jeweiligen Bedienungsanleitung orientiert.

Die Kapazität der Programmspeicher bewegt sich zwischen 50 und 960 möglicher Anweisungen!

2.5.2. Datenspeicher

In die Datenspeicher wird, um es nochmals deutlich zu machen, ein Wert mit der STO-Taste heineingeschrieben, wobei ein ursprünglich darin enthaltener Wert überschrieben wird, d. h. der alte Wert bei dieser Gelegenheit verloren geht.

Nun gibt es aber Fälle, in denen man gerne den ursprünglichen Speicherinhalt nicht verlieren, sondern einen eben angezeigten Wert zum alten Inhalt hineinaddieren will. Dies kann der Fall sein, wenn man einen Zähler einrichtet und in diesen Speicher, nach einem bestimmten Rhythmus den Wert 1 addieren möchte. Dazu gibt es zusätzliche Speichertasten, bzw. Speichertastenkombinationen.

Weil es hier von Hersteller zu Hersteller Unterschiede gibt, ist bei der Untersuchung des Tastenfelds darauf nicht hingewiesen worden. In einem Fall lautet die Befehlsfolge

1 SUM 0

wobei mit 0 der Speicher mit der Speicheradresse 0 gemeint ist. Im zweiten Fall ist die Eingabe wie folgt zu tätigen:

1 STO + 0.

Hier wird die bekannte Speichertaste im Zusammenhang mit dem Pluszeichen dafür verwendet, um dem Rechner anzuzeigen, daß man in den Speicher 0 die Zahl 1 hineinaddiert haben möchte.

Des weiteren kann es passieren, daß im Verlaufe eines Programmablaufes der Inhalt des Speichers mit der Speicheradresse 5 zu halbieren ist.

Im ersten Fall läuft da so ab:

2 INV PRD 5;

und im zweiten Fall:

2 STO ÷ 5.

Diesen Vorgang nennt man **Speicherarithmetik.** Das heißt, ein Speicher, mit dem dieser Vorgang realisiert werden kann, ist ein Speicher mit Speicherarithmetik. Dies ist deswegen zu betonen, da es Rechner gibt, wo dies nicht möglich ist, d. h. in die, wie anfangs erwähnt, eine Zahl nur abgestellt und von dort wieder abgerufen werden kann. Die Speicherarithmetik ist dann anwendbar, wenn man das Ergebnis nicht wissen muß. Ist dies aber doch der Fall, ist die Rechenoperation in der Anzeige abzuwickeln und das Ergebnis dann im

entsprechenden Speicher abzustellen, oder sie müssen nach der speicherarithmetischen Operation den Inhalt dieses so veränderten Speichers in die Anzeige zurückrufen.

Bevor wir uns weiter mit der Speicherproblematik befassen, wollen wir noch die Tastenfolgen ansehen, die für die Speicherarithmetik benötigt werden.

Addition in einen Speicher
Es soll der Wert 5 in den Speicher 2 hineinaddiert werden:
Variante 1: 5 SUM 2
Variante 2: 5 STO + 2

Subtraktion in einen Speicher
Es möge der Wert 2,5 vom Inhalt des Speichers 3 abgezogen werden:
Variante 1: 2,5 INV SUM 3
Variante 2: 2,5 STO – 3

Multiplikation in einen Speicher
Der Inhalt des Speichers 4 möge mit 0,5 multipliziert werden:
Variante 1: 0,5 PRD 4
Variante 2: 0,5 STO x 4

Division in einen Speicher
Der Inhalt des Speichers 7 soll durch 5 geteilt werden:
Variante 1: 5 INV PRD 7
Variante 2: 5 STO ÷ 7

Beachten Sie bitte, daß die Zahl nach dem arithmetischen Befehl immer die Nummer des angesprochenen Speichers ist, d. h. die Speicheradresse. Erinnern Sie sich bitte an die bereits im Abschnitt 2.3.6. besprochene INV-Taste bei Variante 1. Wir haben gehört, daß diese Taste die Umkehrung einer Funktion bewirkt und die Umkehrung der Addition ist naturgemäß die Subtraktion und die der Multiplikation die Division.

Nach Angaben der Hersteller wird für einen Datenspeicherplatz acht mal so viel Raum benötigt, als wie für einen Programmspeicherplatz. Daher ist ein Hersteller (s. S. 120/7) dazu übergegangen, die Aufteilung des vorhandenen frei verfügbaren Speicherangebotes zwischen Daten- und Programmspeichern dem Benutzer zu überlassen. Er kann für jedes Problem eine andere Teilung wählen. Rechenintensive Programme benötigen im allgemeinen mehr Programmspeicherplatz, Probleme des kaufmännisch-betriebswirtschaftlichen Bereiches benötigen eher mehr Datenspeicher.

Wie wir sehen, ist das Angebot an Datenspeichern im allgemeinen aber nicht gerade reichlich. Wir wollen aber nicht unbescheiden sein und uns freuen, daß in einem so kleinen PROTAR überhaupt so viele Datenspeicher untergebracht werden können. Wir müssen zur Kenntnis nehmen, daß auch PROTARe ihre Grenzen haben. Da wir aber wissen, welche Möglichkeiten sich bieten, schicken wir uns an, unserem PROTAR ein echtes Schnippchen zu schlagen, wobei er gar nicht merkt, daß sein Speicherangebot plötzlich ein vielfaches von dem beträgt, was uns unser Lieferant im Prospekt, bzw. der Bedienungsanleitung maximal anzubieten in der Lage war. Wie wir das ausführen wollen, wird im nachfolgenden Abschnitt zu behandeln sein.

2.5.3. Erweiterung des Datenspeicherbereiches

Nachdem also zur Kenntnis genommen werden mußte, daß aufgrund der Konstitution von PROTAR die Speicherungsmöglichkeiten äußerst beschränkt sind, müssen wir uns nach der Decke strecken, vor allem dann, wenn wir uns aus verschiedenen Gründen für einen Rechner entschieden haben, dessen übrige Vorzüge den Nachteil des unzureichenden Speicherplatzangebotes bei weitem wett machen. Daher wollen wir uns in einem eigenen Abschnitt den Möglichkeiten der Erweiterung des Speicherbereiches widmen. Damit glaube

ich besonders jenem Benutzerkreis zu helfen, der aus dem kaufmännisch-betriebswirtschaftlichen Bereich Aufgaben zu lösen hat, an den sich, ohne die übrigen Interessenten zu vernachlässigen, das vorliegende Buch in der Hauptsache wendet.

Die folgenden Darlegungen betreffen einen Rechner der gehobenen Mittelklasse der PROTARe mit einer 10-stelligen Anzeige.

Wir wollen uns von den großen Brüdern der DV einen Begriff ausborgen, und zwar den der „fixen Wortlänge". Den Begriff „Wort" dürfen wir hier aber nicht wörtlich nehmen. Damit ist nur gemeint, daß für bestimmte Speicherarten fixe Größen vorgesehen sind, und zwar fixe Plätze auf den magnetischen Speichermedien. Mit einem derartigen Phänomen haben wir es auch hier zu tun. Im Gegensatz dazu gibt es Rechenanlagen mit sogenannter „variabler Wortlänge", die Speicher haben, deren Größe sich nach der notwendigen Stellenzahl einer Zahl richtet. Aber eine derartige Einrichtung ist nur größeren Rechnern vorbehalten.

Es gibt nun eine Möglichkeit, diese „fixe Wortlänge" zu umgehen. Dies ist dann möglich, wenn die zu speichernde Zahl eine geringere Stellenzahl hat, als die mit zehn Stellen angenommene Speicherkapazität unseres PROTARs. Der Trick liegt nun darin, daß wir zwar hiefür allerdings etwas Programmspeicherplatz benötigen, was wir in unserem Fall wahrscheinlich leichter ermöglichen können, und eine Speicherteilung mit Hilfe eines Programmzweiges realisieren. Wie schon an anderer Stelle angedeutet, benötigen wir allerdings einen PROTAR, der auf seinem Tastenfeld die Tasten INT (ganzzahliger Teil einer Zahl) und INV mit INT, bzw. FRAC (Dezimalteil einer Zahl) aufzuweisen hat; das ist Voraussetzung. Mit diesen beiden Tasten sind meines Wissen aber in der letzten Zeit nahezu sämtliche PROTARe ausgestattet worden. Wir könnten so beispielsweise fünf zweistellige, oder zwei fünfstellige Zahlen in einem Speicher unterbringen. Auch bei diesem Trick müssen wir den Dezimalpunkt mehrmals verschieben. Diesen Vorgang haben wir ebenfalls schon im Abschnitt 2.3.2. kennengelernt, und zwar mit der EE-, bzw. EEX-Taste. Mit Hilfe dieser vorgenannten Elemente sind wir nun in der Lage, jeden Speicherinhalt in einzelne Bestandteile zu zerlegen.

Diese Technik sei an einem Beispiel demonstriert:

Wir nehmen 2 fünfstellige Zahlen, die der Einfachheit alber mit AAAAA und BBBBB bezeichnet werden sollen.

Die Wahl von Buchstaben anstelle von Ziffern wurde bewußt gewählt. Durch diese Verschlüsselung – die ja am Rechner nicht realisierbar ist – soll die freie Wahl von Zahlen angedeutet werden. In einer Formel setzen wir auch Buchstaben anstelle von Zahlen, um damit die universelle Verwendbarkeit dieser Formel zu dokumentieren. Das gleiche gilt selbstredend auch für den Ablauf eines Programms. Denn im Grunde ist ein Programm nichts anderes, als eine elektronisch gespeicherte Formel, die jederzeit benutzt werden kann. Die Buchstaben sind gewissermaßen die Platzhalter für Zahlen.

Als Darstellung in einem Speicher können wir grundsätzlich aus drei Varianten wählen:

1. AAAAABBBBB. 2. AAAAA.BBBBB 3. .AAAAABBBBB

Es sei noch vermerkt, daß der Dezimalpunkt keine eigene Stelle einnimmt, d. h. mit anderen Worten, daß er beim Zählen der Stellen nicht mitgezählt wird, sowie es rein textlich der Fall sein müßte.

Uns scheint die zweite Variante für den Fall, daß wir es mit zwei Zahlen zu tun haben, die geeignetste zu sein. Zuerst wollen wir den einfacheren Fall des Abrufes einer der beiden Zahlen diskutieren.

AAAAA und BBBBB seien zwei fünfstellige ganze Zahlen, die aus der Nachbarschaft der jeweils anderen Zahl herausgelöst werden sollen, damit mit ihnen weitergearbeitet werden kann.

Vorerst wollen wir mit Worten versuchen zu beschreiben, was wir im einzelnen Fall zu tun haben und dann werden wir auch die Befehlsfoge in den beiden Logikarten, in AL, bzw. UPN anschreiben; es werde hier der Speicher 7 verwendet.

Herauslösen der Zahl AAAAA

- Abruf des Speicherinhalts AAAAA.BBBBB in den Anzeigespeicher mit der Taste RCL.
- Drücken der Taste, mit der der ganzzahlige Teil einer Zahl berechnet wird.

Befehlsfolge in AL

RCL 7	Anzeige AAAAA.BBBBB
INT	Anzeige AAAAA.

Befehlsfolge in UPN

RCL 7	Anzeige AAAAA.BBBBB
INT	Anzeige AAAAA.

Wir können keinen Unterschied erkennen; AAAAA ist isoliert und kann somit weiter verarbeitet werden.

Herauslösen der Zahl BBBBB

- Abruf des Speicherinhalts in das Anzeigeregister.
- Berechnung des Dezimalteiles.
- Multiplikation mit 100.000.

Anstelle der Multiplikation mit der Zahl 100.000 hätten wir eine bessere, weil kürzere, Lösung wählen können. Es wird auf den Abschnitt 2.3.2. verwiesen, wo diese Lösung bei der Besprechung der Tasten 10^x, EE und EEX an Beispielen erläutert wurde.Die Zahl 100.000 wird hier aus Platzgründen nicht wie in einem Programm ziffernweise untereinander geschrieben, sondern normal dargestellt.

Befehlsfolge in AL

RCL 7	Anzeige AAAAA.BBBBB
INV INT	Anzeige .BBBBB
x	Anzeige .BBBBB
100.000	Anzeige 100000.
=	Anzeige BBBBB.

Befehlsfolge in UPN

RCL 7	Anzeige AAAAA.BBBBB
FRAC	Anzeige .BBBBB
100.000	Anzeige 100000.
x	Anzeige BBBBB.

Womit auch die Zahl BBBBB isoliert ist und mit ihr weitergearbeitet werden kann. Die etwas schwierigere Frage ist nun die, wie wir wohl diese beiden Zahlen unabhängig voneinander aber miteinander abspeichern können. Dazu wollen wir noch eine Forderung stellen, daß wir bei der Abspeicherung eine allenfalls an der jeweils gleichen Stelle im Speicher stehende alte Zahl überschreiben wollen. Mit anderen Worten bedeutet dieser Wunsch, daß wir sowohl AAAAA als auch BBBBB in ihren Größen beliebig tauschen können, ohne daß die ursprüngliche Zahl vorher gelöscht werden muß.

Abspeichern der Zahl AAAAA in den linken Speicherteil

- Eingabe AAAAA in das Anzeigeregister.
- Abruf des Speichers 7.
- Berechnung des Dezimalteils des Inhalts von Speicher 7.
- Addition AAAAA mit dem Dezimalteil von Speicher 7;
- Abstellen des Ergebnisses in Speicher 7.

Befehlsfolge in AL (Wert A_{alt} sei hier C)

AAAAA	Anzeige AAAAA
+	Anzeige AAAAA.
RCL 7	Anzeige CCCCC.BBBBB
INV INT	Anzeige .BBBBB
=	Anzeige AAAAA.BBBBB
STO 7	

Befehlsfolge in UPN

AAAAA	Anzeige AAAAA
RCL 7	Anzeige CCCCC.BBBBB (automatischer Stack-Lift!)
FRAC	Anzeige .BBBBB
+	Anzeige AAAAA.BBBBB
STO 7	

Abspeichern der Zahl BBBBB in den rechten Speicherteil (Inhalt des Speichers 7 sei AAAAA.CCCCC)

- Eingabe BBBBB in das Anzeigeregister.
- Division durch 100.000.
- Abruf des Inhalts des Speichers 7.
- Berechnung des ganzzahligen Teils dieses Wertes.
- Addition der beiden letztgenannten Werte.
- Abstellen des Ergebnisses in Speicher 7.

Befehlsfolge in AL

BBBBB	Anzeige BBBBB
÷	Anzeige BBBBB.
100.000	Anzeige 100000.
+	Anzeige .BBBBB
RCL 7	Anzeige AAAAA.CCCCC
INT	Anzeige AAAAA.
=	Anzeige AAAAA.BBBBB
STO 7	

Befehlsfolge in UPN

BBBBB	Anzeige BBBBB
ENTER	Anzeige BBBBB.
100.000	Anzeige 100000.
÷	Anzeige .BBBBB
RCL 7	Anzeige AAAAA.CCCCC
INT	Anzeige AAAAA.
+	Anzeige AAAAA.BBBBB
STO 7	

Unter Berücksichtigung der Erläuterungen zu den Funktionstasten INT und FRAC, bzw. INV INT läßt sich in den einzelnen Schritten genau verfolgen, was passiert. Sollten wir uns in den einzelnen Schritten nicht ganz sicher sein, überschlagen wir nochmals kurz den Abschnitt 2.2., in dem die einzelnen Grundrechnungsarten für beide Logikformen diskutiert wurden. Das Abrufen eines Speicherinhalts entspricht dem Vorgang der Eingabe einer Zahl. Zu UPN ist noch zu bemerken, daß mit dem Abruf eines Speicherinhalts nicht nur die Eingabe einer Zahl automatisch abgeschlossen ist, sondern daß bei dieser Gelegenheit nicht die ENTER-Taste zu drücken ist, um die Anzeige zur Eingabe einer neuen Zahl frei zu machen, sondern daß hier automatisch ein Stack-Lift erfolgt, sodaß die nächste Operation ausgeführt werden kann.

Es ist übrigens ratsam, vor der Entscheidung für den Kauf eines hochwertigen Rechners zunächst Erfahrungen auf einem einfachen TAR oder PROTAR zu sammeln. Um Sicherheit zu gewinnen, können die hier besprochenen und noch zu besprechenden Beispiele nachvollzogen werden. Bei entsprechender Übung kann man sich dann auch in der Lösung der persönlich am häufigsten anfallenden Aufgaben versuchen. Mit dieser Erfahrung ist die besondere Eignung qualifizierter PROTARe wesentlich leichter zu ermitteln.

Die Eingabe der beiden Zahlen AAAAA und BBBBB in einen leeren Speicher würde anders auch möglich sein, und zwar wie folgt dargestellt:

- Eingabe AAAAA.
- Abspeicherung in Speicher 7 (muß vorher nicht unbedingt gelöscht worden sein).
- Eingabe BBBBB.
- Division durch 100.000.
- Aufsummierung in Speicher 7.

Diese oder ähnliche Methoden haben den Nachteil, daß bei Änderung einer der beiden Zahlen immer auch die andere nicht zu verändernde neu einzugeben ist.

Im Prinzip gleich wäre Abspeicherung und Abruf einer zweistelligen Zahl.

Zur Übung wollen wir noch zwei Beispiele anschließen.

Es sei eine Reihe von fünf zweistelligen Zahlen im Speicher 3 wie folgt dargestellt:

.AABBCCDDEE

Im ersten Fall wollen wir die Zahl CC isolieren und im zweiten die Zahl EE durch die Zahl FF ersetzen.

Isolierung der Zahl CC

- Abruf des Speichers 3 in das Anzeigenregister.
- Multiplikation mit 10.000; damit rutscht der Dezimalpunkt zwischen BB und CC.
- Berechnung des Dezimalteils; es verbleibt .CCDDEE in der Anzeige.
- Multiplikation mit 100; der Dezimalpunkt kommt nun zwischen CC und DD zu liegen.
- Berechnung des ganzzahligen Teils; somit ist CC isoliert, und kann als selbständige Zahl in den weiteren Berechnungen verwendet werden.

Abspeicherung der Zahl FF anstelle EE

- Eingabe der Zahl FF.
- Division durch 100; Anzeige .FF.
- Abruf des Inhalts des Speichers 3 in das Anzeigeregister.
- Multiplikation mit 100,000.00; der Dezimalpunkt liegt nun zwischen den Zahlen DD und EE.
- Berechnung des ganzzahligen Teils dieses Inhalts AABBCCDD.
- Addition beider Werte AABBCCDD. und .FF.
- Division durch 100,000.000;
- Abstellen des Ergebnisses .AABBCCDDFF in den Speicher 3.

Damit mit diesen Beispielen begonnen werden kann, sind die ursprünglichen Zahlen Ihrer ersten Darstellung entsprechend einzutasten und in dem angegebenen Speicher abzustellen.

Man wäre auch in diesem Zusammenhang versucht, sich wieder einen Fachausdruck „auszuleihen" und von einer „gepackten" Darstellung von Zahlen bei PROTARen zu sprechen. Dieses Packen ist im Prinzip bei allen Rechnern möglich, die die vorgenannten Voraussetzungen mitbringen. In der Tat haben wir mehrere Zahlen miteinander „verpackt".

Der richtige Inhalt des Begriffes „verpacken" stimmt sicherlich nicht mit unserer Begriffsverwendung überein, aber die Wirkung ist die gleiche; wir haben für mehrere Zahlen **einen einzigen** Unterschlupf gefunden.

Nun zur Stellung des Dezimalpunktes: Diesen setzen wir so, daß nach Möglichkeit gleichartige Zerlegungs- und Packroutinen, d. h. Programmteile zur Zerlegung bzw. fürs Packen einer Zahl, verwendet werden können. In den späteren Ausführungen werden wir vom Begriff „Unterprogramm" hören. Und solche Unterprogramme sollten Zerlegung und Packen durchlaufen. Wenn wir diese Programmteile so aufbauen, brauchen wir wenig Programmplatz und haben für eine geringe Einbuße an Programmspeichern viele zusätzliche Datenspeichermöglichkeiten gewonnen. Diese Speichertechnik ermöglichte es in einem Beispiel – siehe Anhang A – eine Produktkalkulation aufzubauen, in der bei insgesamt 26 Speicherplätzen, wovon einige für die Verarbeitung reserviert werden mußten, 40 Verrechnungssätze mit 40 Variatoren in 20 Speichern unterzubringen waren. Die Verrechnungsätze waren dreistellig, hatten also nur drei wertführende Ziffern, was ausreicht und zweistellige Variatoren. Dadurch, daß die Verrechnungssätze mit nur drei wertführenden Ziffern gewählt wurden, ergab sich die Notwendigkeit, die jeweils einzutastende Menge in ihrer Mengeneinheit diesem Umstand anzupassen. Diese Mengeneinheit schwankt nun zwischen 1 und 1.000, je nach Wert der Kostenart. Das wird verständlich, wenn man bedenkt, daß in der Darstellung

SSSVV.SSSVV,

wo der Dezimalpunkt die Aufgabe hat, vier Zahlen voneinander zu trennen, ein weiterer Dezimalpunkt keine Berücksichtigung finden könnte. SSS bedeutet der Verrechnungssatz, VV der Variator. Diesen weiteren Dezimalpunkt würden wir aber benötigen, um Verrechnungssätze mit niedrigen Werten pro Mengeneinheit im Speicher darstellen zu können.

Diese Möglichkeiten wurden deswegen so ausführlich erläutert, weil vor allem bei kommerziellen Aufgaben oft ein hoher Speicherbedarf besteht und dieser auf die dargestellte Art und Weise in den meisten anstehenden Fällen durchaus befriedigt werden kann. Ein wenig Übung und Sie sind Meister in der Nutzung von Datenspeichern!

2.5.4. Sonstige Speicher

Neben den Programm- und Datenspeichern gibt es die bei TARen ebenso vorhandenen **Arbeitsspeicher**, in dem die Rechenoperationen durchgeführt werden.

Bei der AL haben wir es nach außen sichtbar mit zwei Arbeitsspeichern zu tun, äußerlich sichtbar dann, wenn eine Speichertauschtaste X ≷ Y im Tastenfeld zu finden ist. Nun, wenn wir mit Klammern arbeiten, müssen zusätzliche Arbeitsspeicher vorhanden sein, um die Zwischenergebnisse der Klammerrechnungen aufnehmen zu können. Eigenartigerweise fehlt in den Bedienungsanleitungen jeder Hinweis darauf; nun ist diese Tatsache sicherlich nicht wesentlich für das Wissen um den Rechenablauf, Hauptsache es funktioniert alles so, wie es beschrieben ist. Eines unterscheidet diese Art von Speichern von den vier Arbeitsspeichern in der UPN, nämlich, daß sie nicht untereinander austauschbar sind, d. h. daß sie auch nicht in irgendeiner Form in die Anzeige zurückgerufen werden können. Dies wäre auch bei dieser Art von Rechenlogik nicht sehr sinnvoll, da man zu sehr in den Rechenablauf eingreifen würde und das bei maximal neun möglichen Klammerebenen ein heilloses Durcheinander ergeben würde!

Bei UPN sind die vier Stackspeicher bereits bekannt; deren Inhalte sind zyklisch vertauschbar und wenn man genau aufpaßt, kann der Stack ebenfalls als zusätzlicher Speicherplatz benutzt werden; man muß nur aufpassen, daß bei einem Verschieben der Speicherinhalte nach oben (Stack-Lift) der zu speichernde Wert nicht über den T-Speicher hinaus abstürzt und somit verloren geht. Der bereits erwähnte Last X-Speicher ist ebenfalls aus bereits erwähnten Gründen bekannt. Bei jeder neuen Belegung des X-Speichers, d. h.

der Anzeige, wird der vorherige Wert des X-Speichers im „Last-X-Register" abgestellt; dieser Umstand kann u. U. auch bei kurzfristigen Korrekturen genutzt werden.

Andere Modelle haben für vergleichende Abfragen ein **Test-Register**. Bei einem Vergleich wird der Inhalt des Test-Registers mit dem augenblicklichen Inhalt der Anzeige verglichen und daraufhin das Programm verzweigt. Dieser Vorgang ist bei Rechnern der AL-Logik üblich. Nur ist darauf zu achten, daß dieses Register manchmal auch als normaler Datenspeicher verwendet werden kann. Daher ist vor Einbau eines Vergleichs zu sehen, ob der Test-Wert sich auch tatsächlich im Test-Register befindet.

2.5.5. Mehrfachverwendung von Speichern

Im Abschnitt 2.5.3. haben wir erfahren, wie man die Kapazität von Datenspeichern „strecken" kann. Es gibt aber auch eine andere Vorgangsweise, mit der zusätzliche Speichermöglichkeiten geschaffen werden können.

Fallweise benötigen wir Zwischenspeicher, um einen Wert, den wir nur wenig später wieder und dann nicht mehr brauchen, abstellen und bei Bedarf wieder abrufen, worauf der Speicher anderen Zwecken zur Verfügung steht. Diesen Umstand nutzen wir im späteren Programmablauf für ähnliche Vorgänge. Man muß sich dabei aber völlig sicher sein, daß man bei derartigen Operationen nicht einen Wert löscht, den man schließlich doch noch gebraucht hätte. Es bedarf also einer gewissen Aufmerksamkeit, um einen solchen Unfall zu vermeiden; aber wir können mit Hilfe der geschilderten Vorgangsweise wertvollen Speicherplatz zusätzlich gewinnen.

2.6. Besonderheiten programmierbarer Rechner

Zum Teil haben wir auch schon bisher da und dort, wo es zweckmäßig erschien, uns mit Besonderheiten beschäftigt, die nicht allen Rechnern eigen sind. Es gibt aber unter der Vielfalt der angebotenen Modelle, vor allem unter jenen der gehobenen Leistungs- und Preisklasse, eine Reihe von erwähnenswerten Sondereinrichtungen, die den jeweiligen Anwendungsbereich nochmals erweitern können und für spezielle Einsätze zusätzliche Möglichkeiten eröffnen.

2.6.1. DSZ – der automatische Zähler

DSZ, das ist eigentlich ein Indexspeicher, der aber auch mit normalen Zahlen belegt werden kann. Dem Indexspeicher wird eine bestimmte Zahl eingegeben, beispielsweise um sicherzustellen, daß ein bestimmter Rechenvorgang vom Programm so oft durchlaufen wird, als es die Zahl im Indexspeicher anzeigt. Nun wird nach jedem Durchlauf die DSZ-Funktion betätigt und damit der Inhalt des Indexspeichers um den Wert 1 vermindert. Anschließend wird rechnerintern abgefragt, ob der Inhalt des Indexspeichers schon 0 ist. Ist dies nicht der Fall, wird ein neuer Rechenvorgang eingeleitet. Ist aber der Wert tatsächlich 0, wird an anderer Stelle im Programm fortgefahren. Wir haben es mit einer „Programmschleife" zu tun, einem Begriff, den wir noch später im Abschnitt 3.4.3. ausführlich zu erläutern haben.

2.6.2. Indiziertes Rechnen

Während die DSZ-Taste fast auf allen PROTARen der gehobenen Klasse vertreten sind, gibt es auch die umgekehrte Einrichtung, nämlich die, daß mit dem Drücken einer Taste der Inhalt des Indexspeichers jeweils um den Wert 1 erhöht wird. Diese Taste ist die ISZ-Taste, die aber nicht alle Modelle aufweisen. Nun stoßen wir mit dieser Taste in Verbindung mit dem Indexspeicher auf eine weitere angenehme Eigenschaft von PROTAR, das **indiziertes Rechnen**.

Was heißt eigentlich indiziert rechnen? Am besten, wir stellen das an einem kleinen Beispiel dar:

- Eingabe des Wertes 5 in das Anzeigeregister;
- Abspeicherung dieses Wertes in das Indexregister;
- Abruf des Inhalts des Speichers mit der Adresse 5, wobei nach der RCL-Taste die i-Taste gedrückt wird; dabei wird der Inhalt jenes Speicher aufgerufen, dessen Adresse gleich ist dem Inhalt des Indexspeichers.

Ein weiteres Beispiel: Der Sprungbefehl GTO (i) bedeutet im gleichen Sinne, daß der Programmsprung – man entschuldige den Vorgriff auf Abschnitt 3.4.1. – zu jener Marke erfolgen möge, deren Nummer im Indexregister enthalten ist. Bei einem Modell (s. S. 120/3) ist auch die Anwahl der Anzahl der Dezimalstellen indirekt möglich. Es braucht nicht weiter betont zu werden, daß z. B. bei 10 Marken (Nr. 0 bis 9) oder 20 Speichern (Nr. 0 bis 19) ein Sprung oder Abruf nicht erfolgen kann, wenn im Indexspeicher der Wert 37 gespeichert wäre; in diesem Fall schaltet der Rechner auf „stur" und beginnt zu blinken oder wir lesen in der Anzeige das beziehungsvolle Wort „ERROR". Außerdem gilt selbstredend auch die Umkehrung eines indirekten Abrufs, also die indirekte Speicherung. Hätte beispielsweise im Indexregister der Wert 7 gestanden, so hätte der Befehl STO (i) zur Folge, daß der in der Anzeige befindliche Wert in den Speicher mit der Speicheradresse 7 abgestellt würde. Soweit hier zum Begriff „indiziertes Rechnen". Ebenso ist die bereits besprochene Speicherarithmetik auch für indiziertes Rechnen geeignet.

Wir sehen auch, daß mit dieser Einrichtung Tabellen gebaut, bzw. Tabellenwerte beliebig abgerufen werden können. Diesen Umstand machen wir uns bei einer Kalkulation zu Nutze, indem wir die Verrechnungssätze der einzelnen Kostenarten als Tabelle in unseren Rechner abstellen, um sie dann wahlweise bei einer Berechnung aufrufen und verwenden zu können. Wir brauchen nicht mehr den Verrechnungswert selbst einzugeben. Es genügt, wenn der Index eingegeben wird sowie die jeweilige Menge, um diesen Kostenanteil berechnen zu können. Bei Kalkulationen wird man allerdings den Index jeweils automatisch um 1 erhöhen und ihn anzeigen lassen, worauf wir gar nur mehr die der angezeigten Kostenart zugehörige Menge eingeben müssen. Nur wenn mehrere Kostenarten nicht benötigt werden, können wir den Index von Hand aus eingeben, um damit den Abruf der nächsten in der Kalkulation benötigten Kostenart auf diese Weise zeitsparender sicher zu stellen. Darüber lesen Sie mehr im Beispiel A im Anhang.

In einigen Modellen (s. S. 120/7) sind auch mehrere Indexspeicher vorgesehen; ich habe diese allerdings im kaufmännisch-betriebswirtschaftlichen Bereich noch nie entbehrt; für technisch-mathematische Belange (Matritzenrechnungen!) haben mehrere Indexspeicher allerdings ihre Bedeutung.

Das indizierte Rechnen erweitert den Anwendungsbereich von PROTARen beträchtlich. Deshalb sollte man dieser Einrichtung erhöhte Aufmerksamkeit widmen. Aber schon der Begriff „Indiziertes Rechnen" bereitet Kopfzerbrechen. Dabei ist diese Art von Rechnen durchaus nicht so kompliziert, wie es von der Begriffsauslegung scheinen mag.
Hierzu folgendes Übungsbeispiel, evtl. für die Erprobung auf einem Leihgerät:

- Speichern Sie die Zahl 3 in den Indexspeicher.
- Stellen Sie den Wert 27 in den Speicher 3.
- Löschen Sie die Anzeige.
- Addieren Sie 13 mit der in dem Speicher 3 abgestellten Zahl.

Diese Addition der 13 können Sie nach AL wie folgt machen:

- Eingabe der Zahl 13 in das Anzeigeregister.
- Drücken der +-Taste.
- Abruf des Speicherinhalts 3.
- Drücken der Gleichheitstaste; Anzeige 40.

Das gleiche Problem wollen wir nun indiziert lösen:

- Eingabe der Zahl 13 in das Anzeigeregister.
- Drücken der Plus-Taste.
- Der Abruf RCL (i) entspricht ebenfalls dem Abruf des Speichers 3. Die Zahl 3 wird je hier als Speicheradresse angesehen.
- Drücken der Gleichheitstaste; die Zahl 40 wird angezeigt.

2.6.3. Flags

Ein „Flag" oder eine „Fahne" wird als eine Art Wegweiser eingesetzt. Ein Flag ist ein Richtungsanzeiger, der in der Lage ist, zwei Richtungen anzugeben, in die ein Programm fortzusetzen ist. Leider müssen wir auch hier wieder ein wenig vorgreifen: Im Prinzip handelt es sich bei den Flags um Programmeinrichtungen, die wir im Abschnitt 3.4. Verzweigungen kennen lernen werden. Eine Verzweigung, das sei in diesem Rahmen vorweg genommen, ist eine Weiche im Programm. Die Weiche selbst wird gestellt durch Bejahung, bzw. Verneinung einer Frage; d. h. also durch die Beantwortung. Wie die Fortsetzung im Programm selbst aussieht, wird dann ausführlich in dem genannten Abschnitt behandelt werden.

Ein Flag, oder eine Fahne, kann gesetzt sein, oder nicht. Hier kann die Abfrage auf zwei Arten gestellt werden: Die Frage kann lauten, ob der Flag gesetzt ist oder nicht, bzw. ob der Flag nicht gesetzt oder doch. Das Problem der Verzweigungen gehört meines Erachtens zu den schwierigsten der Programmiertechnik. Ich möchte nicht verhehlen, daß mir in der Hitze des Gefechtes auch ab und zu in der Beurteilung des Wahrheitsgehaltes einer Abfrage ein Fehler unterlaufen kann, vor allem dann, wenn die Frage negativ gestellt wird, und somit das Programm nicht den geplanten Verlauf nimmt. Hier sei nur vermerkt, daß Verzweigungen nicht nur mit den in Abschnitt 2.3.5. diskutierten Vergleichsbefehlen, sondern auch mit Hilfe der Flags konstruiert werden können.

2.6.4. Konstantspeicher

Dieser Begriff ist nicht zu verwechseln mit einem Speicher, der einen Faktor bei einer Multiplikation oder den Divisor bei einer Division konstant halten kann. Diese Eigenschaft ist den TARen eigen und wird bei PROTARen eigentlich gar nicht verlangt, da dieser Vorgang des Konstanthaltens eines Wertes besser im Wege eines Programmes gelöst werden kann.

Eine Firma (s. S. 120/5) hat Geräte auf den Markt gebracht, die es gestatten, auch in der Betriebsart OFF, also bei abgeschalteter Energiezufuhr, die Inhalte der Programm- und Datenspeicher aufrecht zu erhalten. Dabei wird sicherlich zur Aufrechterhaltung der Magnetisierung ein geringes Maß an Batteriestrom verbraucht. Natürlich ist dieser Zustand nicht unbegrenzt aufrecht zu erhalten, aber so lange, bis Sie wieder an Ihr Problem gehen, wird dieser Zustand sicher anhalten. Diese Einrichtung, „Constant Memory" genannt – im Englischen wird die Bezeichnung eher dem Begriffsinhalt gerecht, als die im Titel dieses Abschnittes gewählte deutsche Übersetzung – ist sicherlich nicht für PROTARe mit einer Magnetstreifeneinrichtung gedacht. Diese Art von Modellen können ja Daten und Programme extern abspeichern und würden daher den Konstantspeicher gar nicht benötigen. Außerdem würde der Batteriestromverbrauch bei einem größeren Gerät verhältnismäßig hoch sein, sodaß sich auch aus diesem Grund für anspruchsvollere Modelle ein „Constant Memory" nicht rentieren dürfte. Es gibt sicher Fälle, in denen Rechner immer wieder zur Lösung ein und desselben Problems benötigt werden, und für solche Lösungen sind PROTARe mit einem Konstantspeicher die geeigneten Geräte.

2.6.5. LAST X-Register

Im Rahmen der Diskussion über Besonderheiten programmierbarer Rechner sei nochmals auf den bereits erwähnten Sonder-Speicher eingegangen. Wie wir gehört haben, ist dies ein Speicher, in dem der Wert der Anzeige, der vor der letzten Veränderung bestanden hat, solange erhalten bleibt, bis die nächste Veränderung den neuen Wert der Anzeige, bzw. des X-Registers in diesem Sonder-Speicher abstellt und den letzten Wert damit löscht.

Durch die Betätigung der LAST X-Taste wird der letzte Wert wieder in die Anzeige zurückgeholt. In disem einzigen Fall gilt die genannte Regel nicht, daß der aktuelle Wert des X-Registers im LAST X-Register abgestellt wird, wenn eine Veränderung der Anzeige vor sich geht. Er kann gar nicht in diesen Speicher gebracht werden, weil das elektrisch gar nicht möglich ist. Mit der Betätigung der LAST X-Taste wird die Übertragung in diesen Speicher gesperrt, denn ich hole ja den letzten Wert – last X! – von dort heraus. Mit anderen Worten, mit dieser Taste kann der Informationsfluß nicht in beiden Richtungen gleichzeitig ablaufen,so wie wir das bei einer Speichertauschtaste kennen gelernt haben. Damit bleibt ein Wert solange in diesem Sonder-Speicher, bis der Übertragungskanal zur neuerlichen Überschreibung eines Wertes aus der Anzeige, bedingt durch einen Rechenvorgang, frei gegeben wird. Diese Einrichtung hat ein Hersteller, (s. S. 120/1) der im allgemeinen Rechner mit der UPN-Logik baut, in allen mit dieser Logik ausgestatteten Modellen vorgesehen.

2.6.6. Magnetstreifen

Im Rahmen unserer Diskussion der Taschenrechner mit Programmiereinrichtung stellen diese kleinen Streifen, die nur etwas mehr als 1 cm breit und 6 bis 7 cm lang sind, wirkliche Besonderheiten dar! Auf Ihnen können Programme und Daten extern gespeichert werden. Wie dies vor sich geht, sagen Ihnen die Bedienungsanleitungen und ebenso, wie sie wieder in den Rechner eingespeichert werden. Der „Insider" spricht dann vom „Einlesen" von Programmen und Daten.

Weil es hier um ganz spezielle Einzelheiten geht, ist in einigen Fällen auf den jeweiligen Hersteller zu verweisen. Beispielsweise gibt es bei einigen Modellen eines Herstellers (s. S. 120/3) die Möglichkeit, bereits gespeicherte Programme mit Hilfe einer Art Overlaytechnik durch neue Programme teilweise zu überschreiben. Sie haben Programmteile, die Sie für mehrere Fälle gemeinsam benötigen und Sie haben wiederum Programmteile, die Sie wechseln müssen, Sie ersparen sich damit, bestimmte Programmteile, die sich in mehreren Programmen wiederholen, mehrmals schreiben zu müssen.

Ein Rechner (s. S. 120/6) besitzt wiederum eine andere bemerkenswerte Einrichtung: durch eine Wahlmöglichkeit wird anläßlich der Übernahme eines Programms auf den Magnetstreifen dieser gegen ein Auslisten der Programm-Anweisungen gesichert. Dieser Tatsache kann man sich bedienen, wenn man für andere Benutzer Programme schreibt. Man kann sich mit Hilfe dieser Einrichtung gegen Diebstahl geistigen Eigentums schützen.

Die Magnetstreifen können aber auch gegen mehrmaliges Überschreiben der auf ihnen gespeicherten Informationen, also Programmen und Daten gesichert werden. Es ist ärgerlich, wenn man ein Programm oder gespeicherte Daten dadurch vernichtet, in dem man neue Informationen auf die magnetisierbare Schicht bringt. Den ursprünglichen Zustand eines Magnetstreifens wieder herzustellen ist sehr zeitraubend; daher sollte man sich dieser Sicherung bedienen.

2.6.7. Listungen

Darunter verstehen wir nicht Zahlentabellen, die wir anläßlich von Berechnungen produziert haben und nun zu Papier bringen wollen. Hier handelt es sich um Dinge, die wir aus dem Rechner herausholen können, wenn der Rechner an einen Drucker angeschlossen ist oder ein solcher schon in den Rechner selbst eingebaut wurde. Sehr wesentlich ist es, wenn der

Rechner Ihnen die Anweisungen ausdrucken kann, die Sie ihm als Programm eingetastet haben. Damit ergibt sich für Sie eine ausgezeichnete Kontrollmöglichkeit, zu überprüfen, ob Sie auch alles richtig gemacht haben, sowie Sie es zuvor auf Ihrem Programmblatt notiert hatten.

Meist ist es auch möglich, auf einen Tastendruck hin sämtliche Inhalte der Datenspeicher ausgelistet zu erhalten; im Einzelfall – so haben wir gelernt – genügt es, die Tast RCL und eine Speicheradresstaste zu drücken, um den Inhalt eines bestimmten Speichers abzurufen und damit zu kontrollieren.

In einem der neuesten Modelle (s. S. 120/6) ist es sogar möglich, eine Liste sämtlicher verwendeten Programm-Marken mit ihren Positionen im Programm ausgedruckt zu erhalten. Wenn Sie eine Programmstelle besonders kennzeichnen müssen – warum, werden wir im Abschnitt 3.3. sehen – dann ist eine sogenannte Marke im Programm zu setzen. Und mit dieser genannten besonderen Einrichtung können Sie nun kontrollieren, ob Sie nicht eine dieser Marken schon verwendet haben.

2.6.8. Module

Unter diesem Titel bahnt sich eine revolutionierende Neuheit an, die zum Zeitpunkt der Drucklegung erst in einem einzigen Fall (s. S. 120/7) dem Verfasser bekannt ist.

Module sind nichts anderes, als zusätzliche Programmspeicher, die einfach in den Rechner eingesetzt werden, wie eine Batterie. Sie sind etwa 2 x 2 cm^2 groß und können bis zu ca. 5000 Programmschritte speichern. Derzeit ist eine Selbstherstellung leider nicht möglich. Es ist zu wünschen und auch zu erwarten, daß derartige Module auch vom Benutzer eines PROTARs hergestellt werden können. Die 5000 Schritte sind auf mehrere Programme aufgeteilt. Diese Programme können entweder in den Programmspeicher des Rechners geladen werden und stehen dort für Berechnungen zur Verfügung. Sie können aber auch direkt im Modul selbst benutzt werden.

Mit diesen Modulen dürften die Magnetstreifen überflüssig werden, wenn es von den Herstellern dem Benutzer ermöglicht werden kann, seine Programme auf diese Art und Weise zu archivieren.

Was hat diese Einrichtung aber für unsere persönliche Absicht, kleine und kleinste Probleme mit PROTAR bewältigen zu wollen, für eine Bedeutung? Die Möglichkeiten sind schwer abschätzbar; aber es gibt doch viele miteinander verknüpfte Probleme – ich denke in diesem Zusammenhang an die verschiedenen Vorbereitungsarbeiten und die Abwicklung der Kalkulation selbst – auch im kaufmännisch-betriebswirtschaftlichen Bereich, wo der genannte Modul ein Gewinn sein kann.

Zeitermittlungsverfahren für die Herstellung, Rezepturen, das Kalkulationsschema sind alles Dinge, die in einem Modul abgespeichert sein könnten. Ebenso Datenprogramme für eine automatisch ablaufende Kalkulation – siehe Beispiel B im Anhang – würden im Modul aufgenommen werden können.

Ich bin überzeugt, daß diese Module dem fortgeschrittenen Anwender auch bei der Lösung kommerzieller Probleme eine große Hilfe sein können.

2.6.9. In letzter Minute

Sozusagen in letzter Minute vor Drucklegung brachten die Hersteller der bekanntesten PROTARe weitere Besonderheiten auf den Markt, die ich Ihnen nicht vorenthalten möchte.

- Rechner mit der Möglichkeit der externen Speicherung von Programmen und Daten haben im allgemeinen auf einem derartigen Magnetstreifen ein sogenanntes „Diagnoseprogramm", mit dem der Rechner auf seine Funktionstüchtigkeit hin überprüft werden kann. Nun ist man dazu übergegangen, dieses Programm im Rechner fix

einzubauen, (s. S. 120/4). Im Zusammenhang mit der erwähnten Einrichtung ist es möglich, aufgrund bestimmter Anzeigen auf die Ursache eines Gebrechens schließen zu können.

- Dieser Vorgang wurde nun auch auf die Programme des Anwenders ausgeweitet und zwar in der Form, daß im Falle des Erkennens logischer Fehler durch den Rechner, in der Anzeige ein Fehlerhinweis durch Angabe einer Code-Nummer erfolgt. Dadurch wird die Redigierung und die Behebung der Fehlerursache wesentlich erleichert. Diese Einrichtung war schon bei größeren Tischrechnern bekannt; zum ersten Mal ist sie nun auch in Taschenrechnern eingebaut worden.
- Zum Teil auch schon bei einigen TARen vorhanden, wird nun auch bei PROTARen dazu übergegangen, die Leserlichkeit längerer Zahlen durch Setzen von Interpunktionszeichen zu verbessern.

2.6.10. Rechen- und Arbeitsgeschwindigkeit

Für den Anfänger scheint eine mehr oder minder große Arbeitsgeschwindigkeit ohne Belang zu sein. Die Gestaltung dieses Buches sieht aber vor, daß es auch später – also im forgeschrittenen Stadium der Praxis – herangezogen werden kann, um das eine oder andere an Information zusätzlich noch herausholen zu können. Und spätestens zu diesem Zeitpunkt wird die Arbeitsgeschwindigkeit eines Rechners bereits von weit größerer Bedeutung für den Benützer sein, als es am Anfang der Fall ist. Deshalb sollte man sich aber schon bei der Anschaffung eines PROTARs einige Gedanken über die Geschwindigkeiten machen, mit der ein PROTAR bereit ist, uns bei der Lösung unserer Aufgaben zu helfen.

Man wird sich später nicht mehr vorstellen können, wie man die Arbeit ohne PROTAR hat schaffen können und nicht lange darauf wird man es leid, immer auf die Ausgabe eines Ergebnisses durch den Rechner warten zu müssen. Zunächst hatte man die Wartezeit nicht gespürt, da man diese teilweise durch die notwendige Notierung von Zwischenergebnissen überbrückt hatte. Als Geübter hat man alle an sich überflüssigen Nebenarbeiten dem PROTAR zur Bearbeitung überlassen und wundert sich, daß dieser nun nicht schneller arbeitet.

Rechnermäßig sind bestimmte Verarbeitungsgeschwindigkeiten vorgegeben. Um hier bereits eine Vorauswahl treffen zu können, kann ich Ihnen nur empfehlen, sich mit Ihren häufigsten Aufgabenstellungen programmtechnisch zu befassen und sie auf den in die engere Wahl gezogenen Rechnermodellen abzuwickeln. Ich meine, daß dies für einen Anfänger zwar schwer zu realisieren ist, doch vor dem Kauf des nächsten Modells durchaus verwirklicht werden kann.

Zuverlässige Aussagen über Rechengeschwindigkeiten hängen von so vielen Einflüssen ab, die sich der Systematisierung entziehen, daß es Ihnen überlassen bleiben muß, sich hier ein Bild anhand der Sie interessierenden Probleme zu machen.

Einen wesentlichen Einfluß auf die Rechengeschwindigkeit haben Sie aber bei der Gestaltung ihrer Programme. Dazu möchte ich Ihnen einige Tips geben, die Sie berücksichtigen können, sobald Sie einige persönliche Erfahrung in der Programmierung gewonnen haben.

Zeitverzögernd wirkt das häufige Anspringen von Unterprogrammen. Solange Sie genügend Programmspeicherplatz haben, sollten Sie wiederkehrende Befehlsfolgen lieber öfter verwenden, als sie in ein Unterprogramm zu verstauen.

Ist ein Unterprogramm aber nicht zu vermeiden, sollten Sie es so plazieren, daß der Rechner die kürzeste Suchzeit aufwenden muß.

Bei den mit der AL arbeitenden Rechnern bedeutet dies, daß Sie das Unterprogramm an den Programmanfang setzen sollten, weil der Rechner das Aufsuchen der gewünschten Marke grundsätzlich am Programmanfang aufnimmt. Im Falle der mit UPN arbeitenden Rechner, beginnt er ab der Stelle zu suchen, wo sich der Programmzeiger gerade befindet. Daher sollten Sie in diesem Fall Ihr Unterprogramm sofort nach dem Hauptprogramm einschieben.

Hier erkennen Sie auch wieder die bei den Programm-Marken über die Mehrfachverwendung von Labeln gemachten Bemerkungen, daß diese bei UPN möglich ist, eben aus dem genannten Grund, weil der Rechner zyklisch von der augenblicklichen Position aus den geforderten Label aufsucht.

Sie werden bei längerer Benutzung Ihres PROTARs sicherlich auch feststellen, daß Sie anfangs – um überhaupt einmal zu einem funktionierenden Programm zu kommen – kompliziertere Rechengänge verwendet haben. Sie werden lernen, die Probleme einfacher zu lösen und kommen dabei in den Genuß höherer Rechengeschwindigkeiten. Und trotzdem werden Sie das Gefühl nicht los, daß Ihr PROTAR noch schneller sein könnte. Bedenken Sie aber, daß alles relativ ist, so auch die Frage nach der Rechengeschwindigkeit.

Außerdem bin ich überzeugt, daß die Hersteller auch im Bereich der Taschenrechner im Verlaufe der Zeit der gestiegenen Bedeutung von Mini-Computern im Taschenrechnerformat Rechnung tragen werden und auch hier – also in der Frage der Arbeitsgeschwindigkeiten – aus den Geräten noch mehr herausholbar machen werden. Nehmen wir das im Anhang unter B dargestellte Beispiel einer automatischen Kalkulation und unterstellen wir, daß Sie eine größere Anzahl von Kalkulationen abzuwickeln haben. Hier sollten Sie dann sehr wohl sämtliche Möglichkeiten nutzen, die Ihnen von einem Rechnermodell und Ihrer eigenen Programmgestaltung gegeben sind, um die Arbeitsgeschwindigkeit insgesamt zu verbessern. Um Rechner, die in die engere Wahl kommen, auf diese Eigenschaft hin zu untersuchen, sollte man relativ kurze Probleme wählen, allerdings mit einem oftmaligen Durchlaufen einer bestimmten Befehlsfolge. Diese Probleme sind programmtechnisch oftmals keine schwierige Angelegenheit, meist rasch zu schreiben und abzuwickeln. Das oftmalige Durchlaufen einer Befehlsfolge muß so gewählt werden, daß Zeiten mit einer Stopuhr leicht festgestellt werden können.

3. Elemente der Programmierung

Soweit es zweckmäßig schien, wurden in den vorangegangenen Kapiteln erste Hinweise zur Technik der Programmierung gegeben. Hier aber kommen wir nun zum Kern der Sache.

Was ist eigentlich ein Programm?

Im übertragenen Sinn kann als Programm vieles angesprochen werden, so das Programm für den Urlaub oder das Programm einer politischen Partei. Programme aller Art haben eines gemeinsam: Es wird zumindest versucht, den Ablauf nach einem einmal gefaßten **Plan** zu gestalten. Es ist geplant, alles nach bestimmten **Regeln** ablaufen zu lassen. Also sehen wir das gleiche Prinzip vor uns, das auch für ein Computerprogramm letztlich zu gelten hat:

Einen Ablauf nach bestimmten Regeln zu planen. Dennoch besteht ein gravierender Unterschied eines Programms für PROTAR im Vergleich zu den etwas scherzhaft gemeinten Beispielen: Ein Computer hält sich minutiös an die ihm aufgetragenen Anweisungen ohne auch nur ein Jota davon abzuweichen, während die Programmbeispiele aus dem menschlichen Bereich im Grund nie so ablaufen, wie sie geplant wurden, und zwar deswegen, weil der Mensch immer wieder eingreift und auch eingreifen muß. Oder kennen Sie ein Partei- oder ein Reiseprogramm, das nicht nachträglichen Änderungen unterzogen werden mußte?

Ein Computer kann dagegen gar nicht vom einmal vorgezeichneten Weg abweichen, da sich in ihm alles aufgrund von elektrischen Zuständen abspielt; daher ist ein Programm für einen Computer eine ausgesprochen starre Angelegenheit. Für uns ist es also nichts anderes als eine Folge von Rechenanweisungen, wonach ein Problem immer wieder in der genau gleichen ein für allemal festgelegten Art und Weise bearbeitet werden kann. Diese sinnvolle Aneinanderreihung von Anweisungen oder Programmbefehlen kann natürlich geändert werden, indem man einige Operationen entfernt, bzw. andere neu einsetzt; aber im Zustand des Rechnens hat der Benützer keinerlei Einfluß auf das Programm mehr und muß sich daher selbst strikt an die von Ihnen aufgestellten Regeln halten!

Andererseits ist PROTAR nicht ganz so dumm, wie er aufgrund dieser Bemerkungen aussehen mag. Er kann sicherlich nicht wissen, daß Sie eigentlich die Zahl 3 eingeben wollten; statt dessen sind Sie eine Tastenreihe zu hoch angekommen und haben die 6 erwischt. PROTAR rechnet da stur mit der 6 weiter, falls Sie die falsche Eingabe nicht bemerkt haben sollten. Hätten Sie aber beispielsweise bei einem Sprungbefehl vergessen, die anzuspringende Programmstelle mit der Label-Adresse (siehe Abschnitt 3.3.) entsprechend zu markieren, dann zeigt PROTAR, was in ihm steckt: Er bleibt stehen, blinkt oder zeigt Ihnen mit ERROR in der Anzeige an, daß er sich in Ihrem Programm nicht mehr zurecht findet. PROTAR ist also durchaus in der Lage, bestimmte logische Fehler zu erkennen und wie geschildert darauf zu reagieren.

Es ist zweifellos einzusehen, daß für einen Dialog zwischen Mensch und Maschine, wir möchten unsere Arbeit mal so ausdrücken, ein **Formalismus** Grundvoraussetzung ist, den wir **Programmiersprache** nennen wollen, auch im Zusammenhang mit PROTARen. Jede Kommunikation zwischen Wesen ist nun einmal auf einer Sprache aufgebaut, auf Symbolen, deren Inhalte von beiden Seiten in gleicher Weise interpretiert werden. Ein solcher Formalismus war auch bei der Konstruktion der PROTARe nicht zu umgehen; von einer Programmiersprache zu reden, wäre hier wohl etwas vermessen. Die einzelnen Tasten, die speziell zur Programmgestaltung vorgesehen sind, rechtfertigen im engeren Sinne nicht, von einer Programmiersprache für PROTARe zu sprechen. Dazu kommt noch, daß wir es bei den einzelnen Modellen mit verschiedensten Ausstattungen zu tun haben.

Am treffendsten sprechen wir von einer **Tastenprogrammierung,** Auch bei großen Computern muß eine Reihe von Tasten gedrückt werden, um ein Programm in den Rechner zu bekommen, nur sind dort die Regeln wesentlich komplizierter.

PROTARe sind wie kleine Kinder, deren Sprache noch einen geringen Wortschatz aufweist, ohne daß deswegen vom Fehlen einer Sprache die Rede sein kann. Wir werden staunen, mit wie wenig „Worten" und „Begriffen" wir eigentlich auskommen! Diese geringe Anzahl von Regeln, mit der wir es bei PROTAR zu tun haben, ermöglicht es uns, verhältnismäßig rasch mit ihm vertraut zu werden und ihn zur Lösung unserer Aufgaben binnen kurzem einsetzen zu können.

Damit wir uns die Durcharbeitung sämtlicher Elemente der Programmierung erleichtern, wollen wir nun ein Beispiel formulieren, anhand dessen wir die einzelnen Elemente näher betrachten können. Das Beispiel selbst wird natürlich den Anforderungen entsprechend variiert werden müssen; aber das Grundproblem, auf das es uns ankommt, bleibt immer das gleiche.

Es sei wieder ein keiner Vorgriff gestattet: Im nächsten Kapitel 4, in dem wir uns mit dem Programm selbst beschäftigen werden, steht ein Abschnitt im Mittelpunkt: Die Problemanalyse. Im Grunde genommen müssen wir fortlaufend überlegen, wie wir ein Problem angehen, wie wir das Problem zu formulieren haben; wir betreiben also Problemanalyse in Permanenz. Die Untersuchungen, wie wir ein Problem anpacken, um es für den Gebrauch auf PROTAR gefügig zu machen, bedeuten das zentrale Problem unserer Ausführungen überhaupt. Denn im wesentlichen kommt es uns darauf an, immer wieder darüber nachzudenken, was und wie es zu tun ist, um zu den gewünschten Ergebnissen zu kommen. Sämtliche anderen Dinge können irgendwo nachgelesen werden, aber wie ein ganz bestimmtes Problem mit PROTAR oder mit einen anderen Rechner bewältigt werden kann, werden Sie wohl in keinem Lexikon finden. Deshalb wird an passenden Stellen immer wieder soviel zusätzlich erklärt, als es für das Herausarbeiten von Details erforderlich ist, um zu einer umfassenden Problemanalyse zu kommen. Das Grundproblem, das wir fortlaufend erweitern wollen, sei folgendes: Wir hätten es mit einer Reihe von Preisen zu tun, die nach unterschiedlichen Prozentsätzen zu erhöhen sind.

Dies ist sicherlich ein Beispiel, das wir mit einem Elektronenrechner der üblichen Bauart, der wenigstens mit einem Konstantwerk ausgestattet ist, wohl einfacher lösen könnten. Ein Konstantwerk ist eine Einrichtung, mit der bei einem Rechenvorgang ein Operand im Rechner fixiert werden kann, während der zweite frei wählbar ist. Der fixe Operand wäre in unserem Fall ein Erhöhungsprozentsatz. Diesen Wert müßten wir so oft wechseln, als unterschiedliche Prozentsätze gegeben sind. Wenn auch unser Problem im Verlaufe unserer Ausführungen mehrfach erweitert wird, so ist es im Grunde doch klein und daher übersichtlich, daß es gerade deswegen als geeignetes Demonstrationsobjekt erscheint, an dem die verschiedenen Elemente der Programmierung leichter verständlich erläutert werden können.

Bevor wir uns jedoch der rechentechnischen Seite unseres Problems zuwenden, wollen wir uns mit den Symbolen befassen, mit deren Hilfe wir es übersichtlich darstellen können. Wenn wir dann im Abschnitt 4.2 das Ablaufschema besprechen, werden wir die Kenntnis der wichtigsten Symbole brauchen. Dieses Schema ist nichts anderes als die bildhafte Darstellung der Rechenvorgänge.

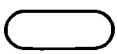
3.1. Symbole

Wofür benötigen wir eigentlich überhaupt die Symbole? Von Symbolen haben wir ja schon gehört, als wir uns mit dem Rechner und seinen Tasten beschäftigten und die Symbole der Tasten kennenlernten. Aber das waren nur Symbole, die dem PROTAR eigen sind.

Wenn wir hier von Symbolen reden, rufen wir uns in Erinnerung, daß wir uns über Elemente der Programmierung unterhalten wollen und das sind wieder ganz andere, vom Rechner völlig unabhängige Zeichen, mit denen wir unser Problem, bzw. dessen Analyse leichter überschaubar darstellen können. Also handelt es sich hier um eine Symbolik, die allein unser Programm und nicht den Rechner angeht.

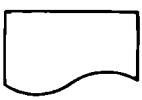
Wenn wir unsere Probleme analysiert haben, stellen wir uns den Programmablauf graphisch dar und müssen natürlich bestimmte Formen wählen, um ein sogenanntes **Ablaufschema** zu erhalten. Wie wir noch später sehen werden, gelingt es mit Hilfe dieser Zeichensprache, ein etwas kompliziertes Problem leichter „in den Griff" zu bekommen. In den meisten Ländern sind diese Symbole genormt. Eine allgemein gültige Symbolik ist vor allem für jene Fälle eine unbedingt notwendige Voraussetzung, wo speziell an umfangreichen, modulartig aufgebauten Standardprogrammen für Anwendungen auf großen Rechenanlagen in verschiedenen Ländern gearbeitet wird. Die Programmierer sprechen ja verschiedene Sprachen und wie schon an anderer Stelle ausdrücklich erwähnt, gibt es in der Elektronik kein „wenn" und kein „aber", es muß alles genau sitzen, jedes Programmdetail muß haarscharf auf das andere passen. Um dieses Ziel zu erreichen, wählt man eine einheitliche Symboldarstellung in den Ablaufplänen.

Für unseren kleinen Bedarf ist die Symbolik nicht von so weitreichender Bedeutung, aber für überflüssig halte ich sie auch wieder nicht. Unnötig ist es allerdings, sämtliche möglichen Symbole zu diskutieren, zumal nur die wenigsten im Zusammenhang mit PROTARen von Bedeutung sind. Und zum zweiten wollen wir hier in der Auslegung, was die einzelnen Symbole bedeuten, auch etwas großzügiger sein. Es kommt weniger darauf an, wie ein Symbol selbst aussieht, sondern welchen Vorgang es darstellt. Dies genügt uns für unsere weitere Arbeit.



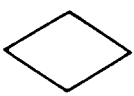
Mit diesem Symbol **begrenzen** wir ein Programm, indem wir die Begriffe BEGINN oder START für den Anfang, bzw. ENDE oder STOP für die Beendigung eintragen.

Dieses Zeichen, an sich sowohl für Ein- und Ausgaben verwendet, wollen wir nur verwenden, wenn wir an einer bestimmten Stelle im Programm eine Zahl **eingeben.**

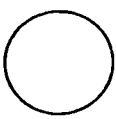


Damit wird ein Schriftstück angedeutet; wir verwenden es, um eine **Ausgabe** eines Ergebnisses darzustellen, gleich ob es sich um eine Ausgabe über einen Drucker oder die Anzeige handelt.

Das Rechteck soll hier einen Rechenvorgang symbolisieren oder allgemein gesprochen eine **Operation**, z.B. auch das Abstellen einer Zahl in einen Speicher.



Die Raute dagegen bezeichnet eine Stelle im Programm, an der wir mit Hilfe einer Abfrage ein Programm **verzweigen** wollen.



Der große Kreis dient als Platzhalter für eine **Programm-Marke**; kann also auch der Beginn eines Unterprogramms sein.

Ein kleiner Kreis dagegen ist nur ein Hilfsmittel, das wir benötigen, wenn ein Ablaufplan an anderer Stelle, z. B. auf einem folgenden Blatt, fortgesetzt werden soll. Er wird daher als **Konnektor** bezeichnet. In diesen Kreis wird irgendein Zeichen, meist eine Zahl notiert, um anzudeuten, wo sich die Fortsetzung befindet; d. h. bei einem Übergang haben wir zwei Konnektoren mit zwei gleichen Zeichen. Wir wissen damit, daß beim ersten Auftreten eine Fortsetzung an anderer Stelle geplant ist. Wenn wir das Zeichen noch einmal finden, kennen wir damit die Stelle der Fortsetzung.

Untereinander sind die Symbole mit Pfeilen zu verbinden, um damit die Richtung der Verarbeitung zu versinnbildlichen. Der Strich verläßt den letzten Vorgang und mündet mit der Spitze beim folgenden.

Es ist anzunehmen, daß dieser Symbolvorrat für unseren Bedarf ausreicht. Wir weichen damit bewußt von der Normung ab. Die Beschränkung auf die hier dargestellten Symbole halte ich nur für Lernzwecke angebracht. Im Zusammenhang mit PROTARen sind auch nur wenige der möglichen Symbole aufgrund der Kleinheit der Rechner vertreten. Außerdem würde hier eine Diskussion sämtlicher Zeichen nur verwirren. In der späteren Praxis wird aber auch die Beachtung der übrigen Sinnbilder empfohlen.

3.2. Normaler Ablauf

Nach diesem geplanten Abweichen in eine kleine Diskussion weiterer Formalismen kehren wir nun zum Grundbeispiel, das wir uns gestellt haben, zurück. Dabei hatten wir uns die Aufgabe gestellt, eine Reihe von Preisen nach unterschiedlichen Prozentsätzen zu erhöhen.

Rein rechentechnisch gesehen gibt es zwei Möglichkeiten, über das Tastenfeld eines Rechners dieses Problem – also von Hand aus und nicht mit Hilfe eines Programmes – zu lösen. Einen Erhöhungsprozentsatz nehmen wir mit 5 an. Die eine Möglichkeit besteht nun darin, jeden eingegebenen Preis mit dem Faktor 1,05 zu multiplizieren. Der Faktor ergibt sich aus folgender Formel:

$$f = 1 + p/100$$

wobei mit p der Erhöhungsprozentsatz gemeint ist.

Wir schreiben an, wie wir uns den Rechenablauf vorstellen:

- Eingabe des alten Preises.
- Drücken der Multiplikations-, bzw. der ENTER-Taste je nach Rechenlogik.
- Eintasten des Erhöhungsfaktors 1,05 und
- Beendigung der Rechnung durch Drücken der Gleichheits-, bzw. Multiplikationstaste.

Im zweiten und vierten Vorgang wurden jeweils zwei mögliche Tastendrücke angegeben. Davon gilt die erste Variante für mit der algebraischen Rechenlogik ausgestattete Rechner, während die zweite auf Rechner mit der umgekehrten polnischen Notation als Rechenlogik zugeschnitten ist.

Haben wir einen Rechner mit einer %-Taste, so können wir auch wie nachfolgend dargestellt vorgehen. Die unterschiedlichen Rechenlogiken gestatten es uns in diesem Fall nicht, eine gemeinsame verbale Beschreibung zu geben.

Zuerst wird daher die Darstellung in AL angeschrieben:

- Eingabe des alten Preises.
- Drücken der Multiplikationstaste.
- Eingabe des Erhöhungsprozentsatzes.
- Drücken der Prozenttaste. In der Anzeige erscheint nun der Betrag, um den der alte Preis erhöht wird.
- Drücken der Gleichheitstaste. Der Betrag der Erhöhung wird dem alten Preis zugeschlagen und der erhöhte Preis angezeigt.

In UPN sieht die Darstellung des Ablaufes so aus:

- Eingabe des alten Preises.
- Drücken der ENTER-Taste.
- Eingabe des Erhöhungsprozentsatzes.
- Drücken der Prozenttaste. Der Betrag der Erhöhung wird angezeigt.
- Drücken der Plus-Taste und Anzeige des neuen erhöhten Preises.

Bevor wir nun unser erstes Beispiel als Programm dem Rechner bekannt geben, müssen wir nun noch einige Bemerkungen zur Technik dieser Programmeingabe machen:

- Wir stellen den Schalter unseres PROTARs oder die entsprechende Taste ein, womit der Rechner angewiesen wird, ab nun vorkommende Tastenoperationen als **Programm-Eingabe** zu deuten und Rechenoperationen nicht auszuführen. Wir nennen diesen Zustand, in dem sich der Rechner nun befindet, **Programm-Modus.** Im Gegensatz dazu sprechen wir vom **Run-Modus,** wenn wir andeuten wollen, daß unser Rechner zur Durchführung von Rechenoperationen von Hand oder vom Programm aus bereit ist.
- Nun können wir das Programm der Logik entsprechend eingeben und zwar genau in der angegebenen Tastenfolge.
- Wir stellen nun den ursprünglichen Modus für den Rechenbetrieb durch erneutes Umschalten wieder her.
- Mit der RST oder RTN-Taste – je nach Rechnermodell – stellen wir den Programmzeiger an den Anfang des Programmes.
- Jetzt kommt der spannendste Augenblick der ersten Zahleneingabe als Test.
- Zum Start unseres ersten Programmes drücken wir nun die Start/Stop-Taste, die R/S-Taste also und setzen damit die Tätigkeit des Rechners in Gang. Mit dieser Taste wird nun PROTAR angewiesen, daß eingetastete Programm mit der ersten Zahl „abzufahren".

Befriedigt sehen wir zu, wie der Rechner zu arbeiten beginnt. Aber plötzlich blinkt er, bzw. läßt in der Anzeige das englische Wort ERROR, also IRRTUM, aufleuchten. Wir hatten uns so darauf gefreut, für die mit 10,00 eingegebene Zahl und mit 5 als Prozentsatz der Erhöhung als Ergebnis den neuen Preis mit 10,50 ausgewiesen zu erhalten! Einen Fehler haben wir offensichtlich gemacht, aber welchen? Wir sehen uns dazu die Tastenfolge noch einmal genau an und merken, daß wir vergessen haben, dem Rechner mitzuteilen, daß wir nach Berechnung des neuen Wertes das Programm eigentlich anhalten wollten; wir müssen also nach Eintasten der Rechenanweisungen als letztes dem Rechner im Programm-Modus den R/S-Befehl erteilen. Es gibt zwar Rechner, die von sich aus nach Einschalten jede Programmspeicherstelle mit einem R/S-Befehl belegen. Das Risiko wollen wir aber nicht eingehen und grundsätzlich jedes Programm ordnungsgemäß mit dem Start/Stop-Befehl abschließen.

Nach Eingabe eines Preises und der Berechnung der Erhöhung ist das Programm jedes Mal wieder in der zuvor beschriebenen Weise an den Anfang zu stellen. Damit ist der Rechner zur Berechnung einer weiteren Preiserhöhung um 5% bereit.

Textlich kann eine **Bedienungsanleitung** für ein Programm aussehen, wie wir den Ablauf soeben beschrieben haben. Nun können wir soviele Preise umrechnen, als wir es wünschen.

Die Tastenfolgen für die einzelnen Logikarten lauten:

Tastenfolge in AL (Multiplikation mit 1,05)

x

1

.

0

5

=

Tastenfolge in UPN (Multiplikation mit 1,05):

1

.

0

5

x

Tastenfolge in UPN (Berechnung mit %-Taste):

5

%

+

Vor allem die erste und dritte Tastenfolge entsprechen im Ablauf der natürlichen Darstellung des Rechenvorganges.

Was wir hier vor uns haben, sind im Grunde genommen bereits Programme, bei denen nur noch Anfang und Ende zu markieren sind. Wie dies erfolgt, werden wir in den übrigen Abschnitten sehen. Eine Möglichkeit des Programmabschlusses haben wir allerdings schon kennengelernt. Mit einer Start/Stop-Anweisung bringen wir den Rechner zum Anhalten. Wir fügen also diese Anweisung der ausgewählten Tastenfolge hinzu.

Weil wir hier nun unsere ersten Programme vorliegen haben, möchte ich nur ganz kurz auf den Programmstart eingehen.

Was passiert, wenn wir das Programm starten? Aus der folgenden Bedienungsanleitung für die Kurzprogramme werden Sie den Vorgang leicht erkennen:

- Alten Preis über das Tastenfeld eingeben.
- Mit dem Drücken der Start/Stop-Taste setzen Sie die Tätigkeit des Rechners ingang; das Programm beginnt bei der ersten Rechenanweisung und endet bei Erreichen der R/S-Taste.

Die Eingabe des Preises ist nicht Teil des Programms, sondern hat nur vorbereitenden Charakter. Mit dem Drücken der R/S-Taste wird nicht nur der Rechner gestartet, sondern ihm gleichzeitig mitgeteilt, daß die Eingabe beendet ist.

3.3. Programm-Marken

Das zuvor beschriebene Kurzbeispiel haben wir nun ausreichend behandelt. Dabei sind wir mit unserer Lösung doch nicht ganz so zufrieden gewesen, weil wir nach jeder durchgeführten Preiserhöhung durch eine eigene Tastenbetätigung das Programm wieder an seinen Anfang stellen mußten. Das war lästig und kostete überdies Zeit. Um diesen Nachteil zu vermeiden, haben die Konstrukteure Programm-Marken eingeführt, im englischen „Label" genannt. Alle PROTARe sind mit dieser Einrichtung ausgestattet, auch Rechner mit maximal 50 Programmschritten. Dies ist deswegen notwendig, weil Programmverzweigungen – siehe Abschnitt 3.4 – sonst nicht realisiert werden können. Ein Label ist nichts anderes als eine Markierung im Programm und besetzt ein bis zwei Programmspeicherstellen, je nach Ausführung des Geräts. Dadurch ist es dem Benutzer möglich, eine bestimmte Stelle im Programm direkt anzusteuern und die Berechnungen von dort an fortzuführen. Im einfachsten Fall ist diese Marke die Zeilennummer des Befehls. Eine Zeilennummer besetzt aber keinen eigenen Programmspeicherplatz. In den meisten Fällen werden diese Marken bei 0 beginnend fortlaufend numeriert. Die größeren Modelle unter den PROTARen besitzen aber heute schon Marken, die mit den Buchstaben A, B, C, D und E, bzw. A', B', C', D', und E' oder a, b, c, d und e bezeichnet sind.

Wie nutzt man nun im Programm diese äußerst vorteilhafte Einrichtung?

Man stellt den Rechner auf den Programm-Modus, drückt zuerst die **Label-Taste** und anschließend die **Label-Adresse,** ähnlich wie wir es schon bei der Benutzung der Datenspeicher mit den Speicheradressen kennengelernt haben. Damit ist die betreffende Programmstelle markiert. Label-Adressen können aber auch andere Zeichen sein, d. h. Tasten, die im Tastenfeld des Rechners zu finden sind. Welche Tasten als Label-Adreßtasten nicht verwendet werden dürfen, sagt Ihnen wieder die Bedienungsanleitung. Diesen Umstand machen wir uns nun – zu unserem Beispiel zurückkehrend – zunutze und fügen den „LBL" und das „A" an die Stelle vor dem ersten Rechenbefehl.

Die Tastenfolgen sehen nun wie folgt aus:

Tastenfolge in AL

LBL A

x

1

.

0

5

=

R/S

Tastenfolge in UPN

entweder

LBL A

1

0

5

x

R/S

oder

LBL A

5

%

+

R/S

Nachdem diese Label-Anweisungen im Programm-Modus dem Rechner eingegeben wurden, stellen wir den Rechner in den Run-Modus zurück und können unsere „Bedienungsanleitung" so formulieren:

- Eingabe des alten Preises.
- Drücken der Label-Taste A.

Mit dieser zusätzlichen Anweisung weiß nun der Rechner, daß er mit dem Drücken der Taste A den Rechenvorgang mit dem eingegebenen Wert vorzunehmen hat. Er sucht nun solange, bis er an irgendeiner Stelle im Programmspeicher diese Marke findet und beginnt an dieser Stelle mit der Arbeit. Er vollzieht sie Schritt für Schritt in einer unvorstellbaren Geschwindigkeit, bis er beim Programm-Stop anhält und das Ergebnis anzeigt.

Sie werden mir zugestehen, daß diese Vorgangsweise bedeutende Vorteile gegenüber der in Abschnitt 3.2. genannten Methode hat. In dieser Form wäre nun unser Programm abspeicherbar, d. h. könnte auf einen Magnetstreifen aufgezeichnet werden. An die Stelle der Label-Adresse A hätten wir ohne weiteres auch eine Zahl, zum Beispiel 0, setzen können. In diesem Fall hätte unser Programm mit der Tastenkombination SBR 0 oder GSB 0 gestartet werden müssen.

Wir können also zwischen **Alpha-Labeln** und **numerischen Labeln** unterscheiden. Außerdem werden die Programm-Marken nicht allein für den Start der Programme, sondern auch für die Realisierung der bereits erwähnten Verzweigungen benötigt.

Numerische Label werden in erster Linie für vom Programm aus anzuspringende Stellen gewählt, während die alphabetischen Label vorrangig als Start-Label für Programme, bzw. Programmteile eingesetzt werden. Von den Rechnern her besteht aber keinerlei Beschränkung in der zuvor angedeuteten Art. Diese Empfehlung über die Verwendung der Label dient in erster Linie der Wahrung der Übersichtlichkeit der Programmgestaltung.

Allgemein gilt, daß Marken nur einmal in einem Programm verwendet werden dürfen. Ein Hersteller (s. S. 120/1) hat in seinen Modellen die Möglichkeit der Mehrfachverwendung von Labeln vorgesehen. Dies ist deshalb möglich, da bei einem Programmsprung der Rechner ab der augenblicklichen Stellung des Programmzeigers das Programm zyklisch nach dem verlangten Label absucht. Er tut dies also in Richtung ansteigender Zeilen-Nummern. Hat

der Rechner den Label bis zur letzten Programmzeile nicht gefunden, setzt er die Suche ab der ersten Anweisung fort. Erst wenn er ihn gefunden hat, wird die Arbeit ab dieser Stelle fortgesetzt. Einschränkend ist zu sagen, daß sie ab jener Stelle forgesetzt wird, an der der gesuchte Label beim zyklischen Suchen **das erste Mal** auftritt.

Es liegt auf der Hand, daß bei einer Mehrfachverwendung von Programm-Maken, die durch das zyklische Suchen ermöglicht wird, peinlich genau zu prüfen ist, ob ein bereits einmal verwendeter Label tatsächlich ein weiteres Mal gesetzt werden darf. Bei genauem Studium des Programmablaufes ist die Mehrfachverwendung ohne Schaden für das Programm durchführbar.

Bei den Rechnern eines anderen Herstellers (s. S. 120/2) wird dagegen grundsätzlich mit der Suche bei der ersten Anweisung begonnen. Daher kann in diesem Fall ein Label tatsächlich nur einmal verwendet werden. Dieser Hersteller hat aus diesem Umstand die Zahl der verwendbaren Label vergrößert, indem er zustäzliche Tasten als Label-Adressen definiert hat.

Aus der Sicht der Bedienungskraft von PROTAR noch ein Wort zu Anordnungen von Alpha-Labeln: Die Tasten A, B, C, D und E liegen in einer Reihe meist oberhalb des numerischen Tastenfeldes. Bei einigen Rechnern besteht nun die Möglichkeit, daß wesentliche Unterschiede im Abstand der Alpha-Tasten zu den Zifferntasten bestehen. So wird man aus Gründen der Zweckmäßigkeit selten benutzte Label in der größten Entfernung vom numerischen Tastenfeld und die sehr häufig verwendeten so nahe als nur möglich anordnen. Benötigt man überdies nur eine geringe Anzahl an Start-Labeln, sollte man neben der am häufigsten verwendeten Programm-Marke keinen weiteren Label dulden, um nicht irrtümlicherweise einen falschen Label zu drücken. Diese Bemerkungen zur Anordnung von Labeln sind für den Anfänger sicherlich nur von äußerst geringer Bedeutung. Der Forgeschrittene wird aber diese Möglichkeiten zu schätzen wissen.

Es geht mir in erster Linie darum, einen weitgehenden Schutz zu gewährleisten, um Eingabefehler möglichst ganz auszuschalten. Dies vor allem deshalb, weil die einzelnen Tasten mitunter sehr klein sind und dazu oft relativ nahe beieinander liegen, sodaß die Betätigung falscher Label-Tasten nie ganz auszuschließen ist. Daher sollte, wie gesagt, auch eine Korrektur möglich sein, um den jeweils letzten Vorgang wieder rückgängig machen zu können. Das ist beispielsweise bei Kalkulationen mit laufender Dateneingabe von Hand sehr wichtig!

Häufig benutzte Label sollten als **Primär-Label** – angesprochen durch die Betätigung der Tasten A, B, C, D und E allein – und weniger häufige als **Sekundär-Label** – angesprochen durch eine Vorwahltaste und eine der genannten Buchstabentasten – gewählt werden.

Man wird aber sehr schnell auf eine optimale Label-Anordnung kommen, wenn man PROTAR einige Zeit lang in der Hand gehabt hat. Die Art der Anordnung von Programm-Marken kann auch von Gerät zu Gerät verschieden sein, sodaß auch unterschiedliche Voraussetzungen für ihre optimale Anordnung gegeben sind. Diese Bemerkungen sollen nur als Anregungen verstanden sein. Es ist darauf hinzuweisen, daß man auch damit Vorteile in der Nutzung von PROTAR erzielen kann.

3.4. Verzweigungen

Ohne Zweifel sind wir nun beim Kernproblem der Programmierung und damit beim schwierigsten Abschnitt angelangt. Diese Schwierigkeit trifft uns aber nicht nur deswegen, weil wir PROTARe programmieren wollen, sondern gilt ganz allgemein für eine Programmierung in jeder Programmiersprache. Eine Verzweigung ist nämlich nicht eine Frage irgendeiner Sprache, sondern die eines Programmablaufes. Mit einer Verzweigung können mehrere Varianten von Lösungen ermöglicht werden. Je mehr Verzweigungen vorgesehen

werden, desto vielfältiger werden auch die Lösungsmöglichkeiten. Aus einem normalen Programmablauf wird ein immer mehr verzweigter und verästelter, weil sich der Rechenvorgang jedes Mal entsprechend der Beantwortung einer Frage, bzw. der Beurteilung einer Vergleichsanweisung, teilt. Allerdings wird es mit jeder Verzweigung schwieriger, die Übersicht zu behalten. Mit zunehmender Praxis aber werden uns die Verzweigungen immer weniger Mühe machen.

Mit Programmverzweigungen lassen sich bei Berücksichtigung der Programmspeicherkapazität unter Umständen mehrere Lösungsmöglichkeiten vorsehen. Ein kleines Beispiel dazu:

Sie müssen unterschiedliche Fortsetzungen von Rechenabläufen vorsehen, je nachdem, ob ein Zwischenergebnis an einer bestimmten Stelle im Programm einen positiven oder negativen Wert angenommen hat. Das wäre dann der Fall, wenn aus einer Zahl eine Quadratwurzel zu ziehen ist. Dieses Problem können wir bequem mit einer Verzweigung lösen. Je nach Art des Vorzeichens des Zwischenergebnisses wird einer von zwei voneinander verschiedenen Rechenvorgängen durchlaufen. Den Schalter für diese Verzweigung stellt die Frage, ob das Zwischenergebnis kleiner als null ist, dar. Den Wert null selbst können wir hier wie einen positiven Wert behandeln.

Eine Verzweigung kann durchaus mit einer Art Signaleinrichtung verglichen werden. Wir wollen ja auch nicht immer unbedingt nur geradeaus fahren, sondern auch mal nach links oder rechts in eine Nebengasse einbiegen, weil wir auch dort etwas zu erledigen haben. Alles kann nicht an der Hauptstraße liegen.

Die Hürde der Verzweigungen zu nehmen, kostet einige Anstrengung. Wem gerade dieser Abschnitt besondere Mühe macht, sei gesagt, daß auch der Autor dieses Buches noch heute als „erfahrener Hase" manches Mal über die Fußangeln einer Verzweigung stolpert. Das wesentliche dabei ist das baldige Erkennen von Fehlern. Bei einem Test, bei dem sämtliche eingebauten Abweichungen geprüft werden sollen, wird man im allgemeinen sehr bald auf derartige Fehler stoßen, besonders leicht dann, wenn auf eine negativ gestellte Farge die Antwort „ja" zu geben ist.

Beispiel: Der Wert in der Anzeige sei 0,5. Folgende Frage sei vorgesehen: Ist der Wert in der Anzeige nicht 0? Würden Sie mit nein oder ja antworten? Bevor Sie die Antwort lesen, decken Sie bitte die folgenden Zeilen ab und versuchen Sie vorerst selbst die richtige Antwort zu finden.

Die Antwort muß richtig lauten: Ja. „Nicht 0" und „Nein" wäre eine doppelte Verneinung und damit eine Bejahung, aber im Hinblick auf einen Wert 0 in der Anzeige! Das als Beispiel, welche Art von Schwierigkeiten Sie zu überwinden haben.

Am wichtigsten ist, die Frage gleich zu Beginn richtig zu formulieren, also die Bedingung zu stellen und andererseits damit auch die Möglichkeiten der Beantwortung richtig einzuschätzen. Mit diesem Spezialproblem wollen wir uns im Kapitel über die bedingten Sprünge, die Verzweigungen im eigentlichen Sinne, noch sehr intensiv befassen.

Wie schon erwähnt, sind diese Programmweichen sozusagen das Salz in der Suppe und bedürfen ein hohes Maß an geistiger Mobilität und der Fähigkeit, einen logischen Ablauf richtig zu erkennen und zu analysieren. Aber gerade diese Fähigkeit läßt sich schulen und zwar am besten damit, daß wir unsere täglichen Abläufe einmal Schritt für Schritt unter die Lupe nehmen. Dabei werden wir feststellen, wo Varianten zu berücksichtigen sind und wie. Wir müssen lernen, uns bei unserer eigenen Arbeit selbst genau auf die Finger zu sehen. In diesem Abschnitt hole ich bewußt weiter aus, um das Problem der Verzweigungen möglichst leicht verständlich aufzuzeigen. Einerseits sollte das wesentliche gesagt werden, andererseits soll durch Langatmigkeit der Kern der Sache nicht verdeckt werden. Daher wollen wir das bereits begonnene Beispiel mit der Aufwertung von Preisen weiter ausbauen.

3.4.1. Unbedingte Sprünge

Als Grundproblem haben wir unser Beispiel so formuliert, daß wir Preise mit verschiedenen Erhöhungssätzen aufrechnen wollten. Einen Prozentsatz haben wir bereits mit 5% angesetzt. Unser Beispiel also erweiternd wollen wir einen zweiten Erhöhungssatz mit 6% annehmen.

Über Marken oder Label haben wir gelernt, daß sie zur Kennzeichnung unterschiedlicher Programmzweige herangezogen werden. Für den zweiten Erhöhungssatz wollen wir daher den Label B wählen. Ich wiederhole somit, daß über den Label A Preise um 5% und über den Label B Preise um 6% erhöht werden sollen. Im Prinzip kann ich nun unter dem Label B die gleiche Befehlsfolge ansetzen, wie wir sie für die Erhöhung um 5% schon verwendet haben; an sich ist zu dieser Lösung nichts zu sagen, solange genügend Programmspeicherplätze zur Verfügung stehen. Da wir aber grundsätzlich daran Mangel leiden, lösen wir unser Problem immer mit Hilfe des geringsten Programmspeicherbedarfes.

Rechentechnisch gesehen kann wie bereits kurz zuvor erwähnt, der gleiche Programmzweig gewählt werden, nur muß gewährleistet sein, daß einmal die Zahl 5, das andere Mal die Zahl 6 als Prozentsatz herangezogen wird. Und wie schon im Titel angedeutet, benötigen wir für dieses Vorhaben einen unbedingten Sprung. Für diese Problemlösung zeigt sich, daß Rechner mit Prozenttaste im Vorteil sind, denn der Einsatz der Multiplikationsformel bringt keinen wesentlichen Vorteil.

Vorerst wollen wir einmal definieren, was ein **unbedingter Sprung** eigentlich ist. Im Verlaufe unseres Programmes kommen wir ohne jede Vorbedingung an eine Stelle, an der ein Sprungbefehl mit einer Betätigung der GTO-Taste markiert wurde. Diese Taste allein bewirkt bekanntlich noch gar nichts, denn sie muß von einer Sprungadresse gefolgt sein. Diese Adresse ist gleichzeitig auch die Label-Adresse und markiert somit jene Stelle, wohin sich der Programmsprung richtet.

Wir erklären vorerst einmal mit Worten, was in unserer jetzigen Programmversion passiert:

1. Eingabe eines mit 5% umzurechnenden Preises über die Marke A.
2. Der Rechner setzt die Zahl 5.
3. Es erfolgt der Sprung zu Label 0.
4. Eingabe eines mit 6% umzurechnenden Preises über die Marke B.
5. Der Rechner setzt die Zahl 6.
6. Hier ist nun der Label 0 gesetzt; an dieser Stelle beginnt nun die gemeinsame Rechenarbeit.
7. Es wird der Wert errechnet, um den der alte Preis zu erhöhen ist.
8. Der neue Preis wird gebildet.
9. Das Programm hält mit der Anzeige des neuen Preises an.

Was wir hier verbal erklärt haben, muß nur noch in Tastenschritte umgesetzt werden. Die Punkte 1 bis 9 sind aber nicht mit einer Bedienungsanleitung für dieses Programm zu verwechseln. Diese müßte dagegen wie folgt aussehen.

1. Eingabe der um 5% zu erhöhenden Preise über den Label A.
2. Eingabe der um 6% zu erhöhenden Preise über den Label B.

Mehr ist zur Lösung dieses Problems in einer Bedienungsanleitung nicht zu sagen. Und nun wollen wir die Tastenschritte wieder in den einzelnen Logikarten darstellen.

Tastenschritte in AL

LBL A	Eingabe der um 5% zu erhöhenden Preise
x	
1	
.	
0	
5	Der Rechner setzt den Erhöhungsfaktor
GTO 0	Sprung zur gemeinsamen Verarbeitung
LBL B	Eingabe der um 6% zu erhöhenden Preise
x	
1	
.	
0	
6	Der Rechner setzt den Erhöhungsfaktor
LBL 0	Hier beginnt die gemeinsame Verarbeitung . . .
=	. . . mit dem Setzen des Gleichheitszeichens
R/S	Das Programm hält mit der Anzeige des neuen Preises an.

Tastenschritte in UPN

LBL A	Eingabe der um 5% zu erhöhenden Preise
1	
.	
0	
5	Der Rechner setzt den Erhöhungsfaktor
GTO 0	Sprung zur gemeinsamen Verarbeitung
LBL B	Eingabe der um 6% zu erhöhenden Preise
1	
.	
0	
6	Der Rechner setzt den Erhöhungsfaktor
LBL 0	Hier beginnt die gemeinsame Verarbeitung
x	Der neue Preis wird berechnet
RTN	Das Programm hält mit der Anzeige des neuen Preises an.

Damit haben wir schon ein komplettes funktionstüchtiges Programm geschrieben, wobei wir uns der Multiplikationsmethode bedienten.

Zunächst wollen wir analysieren, was bei dem zuvor dargestellten Programmablauf passiert.

Fall der Erhöhungen um 5%: Wir geben einen Preis 12,00 und drücken anschließend die Label-Taste A. Der Rechner sucht nach der Programm-Marke A. Er findet sie und einen Schritt weitergehend erkennt er, daß der Faktor 1,05 mitzunehmen ist. Dann muß er an die Programmstelle mit der Marke 0 springen, Preis und Erhöhungsfaktor in den Arbeitsspeichern mitschleppend. Mit der Ausführung der Berechnung und der Anzeige des Ergebnisses 12,60 beendet PROTAR die Arbeit und bleibt stehen.

Fall der Erhöhungen um 6%: Nun wollen wir einen anderen Preis, angenommen 20,00, eingeben. Mit dem Drücken der Label-Taste B starten wir das Programm. Der Rechner beginnt an der Stelle mit seiner Arbeit, an der er die Marke B findet. Von da an nimmt er den Faktor 1,06 mit, ignoriert die Marke 0 und berechnet den neuen Preis im gleichen Programmzweig wie zuvor. PROTAR beendet mit der Anzeige des neuen Preises 21,20 die Verarbeitung und hält an.

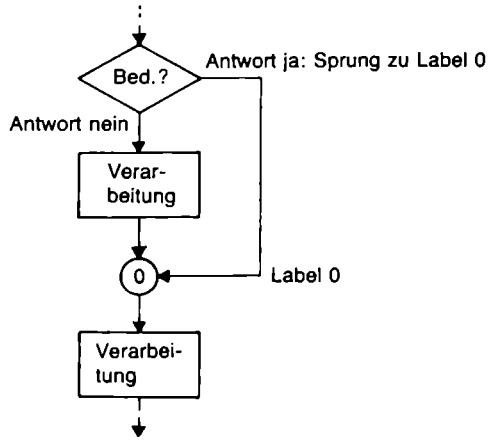
Wir haben im Fall A einen unbedingten Sprung kennengelernt und im Fall B gesehen, daß das Programm einen Label einfach „übersieht". Und das mit Recht! Da das Programm zu diesem Label kommt, ohne irgendwie dazu gezwungen worden zu sein, einfach weil er mitten im Programmablauf steht, ist diese Programmarke für ihn nicht existent. Wenn das Programm im Fall B über diesen Label kommt, ist das wie ein Leerbefehl, an dem einfach nichts passiert.

3.4.2. Bedingte Sprünge

Aufgrund verschiedener Vorgriffe im Thema ist teilweise schon angeklungen, was es mit den bedingten Sprüngen für eine Bewandtnis hat. Ein bedingter Sprung wird dann angesetzt, wenn das Programm angewiesen wird, selbst abzuprüfen, welchen Weg es aufgrund einer gestellten Bedingung gehen soll. Die Grundlage dieser Bedingung kann durch eine Eingabe geschaffen werden – wie wir das in Verfolgung unseres Beispiel vor uns haben – sie kann aber auch erst bei Beurteilung eines Zwischenergebnisses auftreten, beispielsweise wenn eine Programmverzweigung dann eintreten soll, wenn ein Zwischenergebnis an einer bestimmten Stelle negativ ist.

Woher immer auch die Grundlagen für diese Bedingung kommen, wesentlich ist, daß aufgrund des unterschiedlichen Wahrheitsgehaltes das Programm angewiesen wird, eine bestimmte Richtung einzuschlagen.

Zum besseren Verständnis sei eine allgemein gehaltene Abfrage in graphischer Darstellung präsentiert.



In der Tastenfolge sieht das Problem, das mit unserem Übungsbeispiel nichts zu tun hat, etwas anders aus, wobei hier anstelle der einzelnen Tasten eine verbale Beschreibung gesetzt wird:

- Verarbeitung
- **Bedingung?**
- **Sprung nach Label 0** (bei Antwort ja)
- **Verarbeitung** (bei Antwort nein)
- **Label 0**
- Verarbeitung usw.

In diesem Ablauf haben wir die wesentlichen Programmstellen durch Fettdruck gekennzeichnet, während mit dem Text „Verarbeitung" irgendwelche Rechenoperationen allgemeiner Art angedeutet werden.

Eines wollen wir uns dabei merken: Im Falle, **daß die Bedingung zu bejahen ist, wird der nächstfolgende Befehl im Programmablauf ausgeführt. War die Bedingung zu verneinen, wird das Programm automatisch beim übernächsten Befehl fortgesetzt.**

Dabei ist, wie auch in unserem allgemeinen Beispiel, dieser der Bedingung nächstfolgende Befehl meist eine Sprunganweisung, er muß es aber nicht sein. Hier gibt es auch von Rechner zu Rechner Unterschiede; denn bei manchen Rechnern muß dieser Befehl unbedingt eine Sprunganweisung sein; was eine Einschränkung darstellt.

Dieser Satz ist ein **Merksatz**, den wir unbedingt behalten wollen und uns einprägen müssen. Denn er sagt genau, wie die Weichen zu stellen sind. Die in einem Programm vorsehbaren Abfragen wurden ja schon bei der Diskussion des Tastenfeldes im Abschnitt 2.3.5. besonders hervorgehoben.

Auch in der Einleitung zu diesem besonders wichtigen Abschnitt haben wir ja auch schon angedeutet, wie schwierig Fragestellung und Beantwortung mitunter sein können. Lesen Sie bitte im Abschnitt 3.4. noch einmal nach, wo gefragt wurde, ob der Wert einer Anzeige nicht 0 ist!

Nun aber zurück zu unserem **Übungsbeispiel mit den Preiserhöhungen:** Das Beispiel soll dabei weiter variiert werden, indem wir die Forderung aufstellen, daß die Preiserhöhungen um 5, bzw. 6% jetzt von der ursprünglichen Preishöhe abhängig gemacht werden sollen. Liegen alte Preise unter 100,00, sollen sie um 6% und Preise um 100,00 und darüber um 5% erhöht werden. Bei dieser Feststellung ist es besonders wichtig, auch die Lage des Grenzwertes 100,00 genau zu fixieren, welcher Gruppe von Erhöhungen er zugerechnet werden soll. Wenn hier keine Regelung erfolgt, kann es sein, daß der Rechner beim Auftreten des Grenzwertes plötzlich zu blinken beginnt, weil er nicht wissen kann, wo Sie ihn für die Preiserhöhungen einordnen wollten. Hinsichtlich der Festlegung eines Grenzwertes mögen beispielsweise auch andere Überlegungen eine Rolle spielen. Man könnte den Grenzwert mit 98,00 festlegen, weil in der Vergangenheit ein auf 100,00 kalkulierter Preis aus pschologischen Gründen mit 98,00 fixiert wurde. Umgekehrt kann man aber auch davon ausgehen, dem Rechner die Erhöhung der Preise um 5% erst ab 101,00 zu befehlen, weil Preise in der Vergangenheit nicht voll auskalkuliert waren, um die psychologische Grenze von 98,00 oder 100,00 nicht zu überschreiten. In der Fortsetzung des Beispieles setzen wir die Preisgrenze bei 99,99 an. Bis dahin ist mit 6%, ab 100,00 mit 5% zu erhöhen.

Mit dieser Forderung nach unterschiedlichen Preiserhöhungen stellen wir in unserem Beispiel zum ersten Mal eine **Bedingung,** wobei wir nie vergessen dürfen, auch die Frage des Grenzwertes eindeutig zu regeln.

Sie werden sofort erkennen, wie dieser Schalter funktioniert, weil das Rechenbeispiel so einfach und leicht nachvollziehbar ist, Die Vergleichsbefehle haben wir ja schon kennen gelernt.

Zuvor wollen wir den Programmablauf wieder verbal vorbereiten. Dabei stellen wir fest, dass es nunmehr nicht nötig ist, zwei Eingabe-Label anzusetzen, da wir es ja dem Programm überlassen, die Richtung zu bestimmen. So kehren wir wieder allein zum Label A zurück.

1. Eingabe des alten Preises über den Label A.
2. Das Programm setzt den Wert 99,99.
3. Nun folgt die Abfrage, ob der Vergleichswert kleiner oder gleich dem alten Preis ist. Im Falle ja, erfolgt der bedingte Sprung nach Label 1.
4. Im Falle nein, setzt der Rechner die Zahl 6 in die Anzeige und es erfolgt ein unbedingter Sprung nach Label 0.
5. Wir setzen den Label 1.
6. Der Rechner setzt die Zahl 5 in die Anzeige.
7. Wir setzen den Label 0.
8. Nun werden die Werte der Erhöhungen berechnet.
9. Der Rechner hält mit dem jeweils neuen Preis in der Anzeige an.

Nun sehen wir uns die Folge der Tastenbefehle in den beiden Logikarten an. In diesem Fall unterstellen wir, daß PROTAR mit einer %-Taste ausgerüstet ist (s. S. 66).

Wir möchten noch festhalten, daß im Falle des Fehlens einer Prozent-Taste anstelle der Zahlen 5 und 6 die Zahlen 0,05 und 0,06 zu setzen sind, worauf nach dem Label 0 auch das Setzen dieser Taste entfällt.

An dieser Stelle sei auf die Vermeidung der Eingabe überflüssiger Nullen hingewiesen. Die Zahlen 0,05 und 0,06 sind als ,05 und ,06 einzutasten. Dem Dezimalpunkt vorangehende Nullen kosten nur Programmspeicherplatz und haben keinen Nutzeffekt. Ebenso wird man die Eingabe von Nullen nach der letzten wertführenden Ziffer einer Zahl vermeiden, soweit es sich um Ziffern rechts vom Dezimalpunkt handelt.

Diese Bemerkungen haben zweifellos nur für die Erzeugung von Zahlen durch das Programm Bedeutung. Von Hand aus über das Tastenfeld eingegebene Zahlen unterliegen nicht dieser Beschränkung.

Zum Abtesten dieses Programmbeispieles wollen wir nun im Kopf beide Programmzweige prüfen.

Zuerst werde der Preis von 120,00 eingegeben: Der Rechner speichert diesen Preis einmal im sogenannten Testspeicher ab. Dann nimmt er die Zahl 99,99 in die Anzeige und vergleicht diesen Wert mit dem Inhalt des Testspeichers. Er findet, daß die gestellte Bedingung, Anzeige ist kleiner als der Inhalt des Testspeichers und geht nach Label 1. Dort holt er den alten Preis aus dem Testspeicher zurück in die Anzeige und startet die Berechnung der Preiserhöhung um 5%. Dabei übergeht er wieder – wie schon einmal besprochen – den Label 0, da er für ihn bedeutungslos ist und hält dann nach Bildung des neuen Preises an, wobei er uns das Ergebnis mit 126,00 anzeigt.

Daraufhin geben wir ihm einen zweiten Preis ein und zwar 80,00: Der Rechner speichert den Preis wiederum befehlsgemäß im Testspeicher ab. Er nimmt die Zahl 99,99 in die Anzeige und findet im Vergleich der Anzeige mit dem Inhalt des Testspeichers, daß die gestellte Bedingung, Anzeige kleiner oder gleich Inhalt des Testspeichers, diesmal nicht erfüllt ist. Er übergeht daher den Sprungbefehl nach Label 1, holt den alten Preis aus dem Testspeicher zurück in die Anzeige und beginnt die Berechnung der Preiserhöhung zu 6%. Dann macht das Programm den unbedingten Sprung nach Label 0 und berechnet dort den neuen Preis und hält mit der Anzeige 84,80 an.

Tastenfolge in AL

LBL A	Eingabe des alten Preises
STO T	Abspeicherung des Preises im Testspeicher
9	
9	
.	
9	
9	Der Rechner setzt den Grenzwert
x ≤ T	Abfrage nach der Größenordnung des alten Preises
GTO 1	Ist der alte Preis größer als der Grenzwert, wird zur Erhöhung um 5% gesprungen
RCL T	Abruf des alten Preises und Vorbereitung zur Erhöhung um 6%
x	Einleitung der Multiplikation
6	
GTO 0	Sprung zur gemeinsamen Verrechnung
LBL 1	Beginn der Erhöhung um 5%
RCL T	Abruf des alten Preises
x	Einleitung der Multiplikation
5	
LBL 0	Beginn der gemeinsamen Verrechnung
%	Berechnung der Preiserhöhung
+	Einleitung zur Bildung des neuen Preises
RCL T	Abruf des alten Preises
=	Berechnung des neuen Preises
R/S	Stop und Anzeige des neuen Preises

Tastenfolge in UPN

LBL A	Eingabe des alten Preises
9	
9	
.	
9	
9	Der Rechner setzt den Grenzwert als Vergleichszahl
x ≤ y	Der Grenzwert wird mit dem eingetasteten Preis verglichen
GTO 1	Ist der eingetastete Preis größer als der Grenzwert, erfolgt die Erhöhung um 5%
R ↓	Damit wird der eingetastete Preis wieder zurück in die Anzeige geholt
6	Mit dem Setzen der Zahl 6 beginnt die Erhöhung um 6%
GTO 0	Sprung zur Berechnung der Erhöhung und des neuen Preises
LBL 1	Hier beginnt die Erhöhung mit 5%
R ↓	Damit wird der eingetastete Preis wieder zurück in die Anzeige geholt
5	Der Rechner setzt die Zahl 5 in die Anzeige
LBL 0	Beginn der gemeinsamen Verarbeitung
%	Berechnung der Preiserhöhung
+	Der neue Preis wird ermittelt
R/S	Das Programm hält mit der Anzeige des neuen Preises an.

Der Vorgang ist dem in der AL ablaufenden vollkommen gleich. Auch hier wird als Vergleichszahl der Wert 99,99 gesetzt, um anzudeuten, daß ein Preis von genau 100,00 bereits außerhalb der Erhöhung um 6% liegt und daher nur um 5% aufzurechnen ist.

Wir haben also gesehen, daß wir bei dieser Variante unseres Überungsbeispiels mit nur einem Label auskommen, weil wir die Entscheidung, mit welchem Prozentsatz ein eingetasteter Preis zu erhöhen ist, PROTAR überlassen haben. Sicherlich wird der Vorgang dadurch etwas unübersichtlicher; aber wenn man das Programm in Gedanken noch einmal nachvollzieht, bleibt es völlig klar. Der in der UPN-Logik mit der R↓-Taste gesetzte Befehl bewirkt eine Umspeicherung im Stack, womit der eingegebene Preis wieder in den X-Speicher, d. h. in die Anzeige gelangt.

Nun zum Problem der **Abfrage**, bzw. dem **Stellen einer Bedingung:** In unserem Beispiel war gefordert, daß Preise von 100,00 und darüber um 5%, die restlichen um 6% zu erhöhen seien. Wir haben als Grenzwert, den sich der Rechner aufgrund unseres Programms selbst setzt, die Zahl 99,99 gewählt, weil wir die Abfrage nach $x \leq T$, bzw. $x \leq y$ gestellt haben. So hätten wir durchaus als Vergleichswert auch die Zahl 100 in unser Programm einbauen können, nur müßte dann nach $x \geq T$, bzw. $x \geq y$ gefragt werden. Dabei hätten zusätzlich die Stellungen der beiden Zahlen 5 und 6 vertauscht werden müssen. Nun gibt es aber meines Wissens bei PROTARen derzeit diese Art von Abfragen nicht in allen Fällen. Daher hätte die Abfrage nach „größer oder gleich" in die Fragen nach $x=T$, bzw. $x=y$ und nach $x>T$, bzw. $x>y$, je nach der im Rechner verwendeten Rechenlogik, aufgelöst werden müssen. Eine andere Möglichkeit besteht nun darin, die beiden Vergleichszahlen, den Preis und den Grenzwert, platzmäßig zu vertauschen. Dadurch können wir auch die Fragestellung umdrehen und wie folgt fragen: $x<T$, bzw. $x<y$. In der Anzeige x steht nun der alte Preis, und im Testspeicher, bzw. im y-Speicher der Grenzwert. Denn $x \geq T$, bzw. $x \geq y$ ist gleichwertig der Frage $x<T$, bzw. $x<y$ mit vertauschten Werten. Es muß nur sichergestellt werden, daß dann das Setzen der Zahlen 5 und 6 in der richtigen Reihenfolge vorgenommen wird, wie es unserer Problemstellung entspricht, und zwar jener Variante, der wir unsere Forderungen im Bereich der bedingten Sprünge unterlegt hatten.

Sie sehen hier wieder einmal, wie wichtig es ist, die Frage richtig zu formulieren, so daß gewährleistet ist, daß im Falle der Erfüllung der Bedingung immer der nächstfolgende Befehl ausgeführt wird. Und daß im Falle einer Verneinung dieser nächstfolgende Befehl übersprungen und das Programm beim übernächsten Befehl fortgesetzt wird.

Im Kapitel 2.6.3. haben wir von einer besonderen Besonderheit gehört, von den **Flags**. Das sind, wie gesagt, Signale, die gesetzt oder nicht gesetzt sein können. So gehören sie auch hierher in unser laufendes Kapitel, da mit ihnen auf die gleiche Art und Weise bedingte Sprünge ausgeführt werden können. Eine Abfrage nach dem Flag 0 – auch hier beginnt man bei dem Wert 0! – kann wie folgt aussehen:

- Ist Flag 0 gesetzt?
- Ist Flag 0 nicht gesetzt?

Hier gelten genau die gleichen mehrfach besprochenen Regeln, daß im Falle einer Bejahung wiederum der nächste Befehl ausgeführt und im Falle einer Verneinung dieser übersprungen und das Programm beim übernächsten Befehl fortgesetzt wird. Hier ist allerdings zu berücksichtigen, daß die Negativabfrage – ist Flag 0 nicht gesetzt? – nicht bei allen Rechnern möglich ist. In diesem Fall muß man eben die Sprungbefehle anders setzen, um zum gleichen Ergebnis zu kommen.

Es ist wiederum auf die Schwierigkeit einer **Negativabfrage**, hinzuweisen, da die Antwort ja eine heillose Verwirrung stiften könnte, da diese Antwort **ja** gleichbedeutend ist, daß der Flag 0 **nicht** gesetzt ist. Es kann nicht oft genug erwähnt werden, jede Fragestellung und jede Weichenstellung genauest zu überprüfen und abzutesten, da man Wahrheitsgehalt oft mit der Formulierung der Fragen verwechselt. Ich würde also empfehlen, unser Übungsbeispiel auch mit einer anderen Fragestellung, bzw. mit dem angedeuteten anderen Grenzwert neu zu formulieren. Der Grundgehalt einer Frage kann durchaus gleich bleiben. Formuliert man aber die Frage anders, sind auch durch die anderen Antworten die Weichen anders gestellt. Man wird sich aber einerseits nur jene Fragestellungen aussuchen können, deren Beantwortung im Sinne der vergleichenden Programmiertasten ist und andererseits bei mehreren Möglichkeiten jene aussuchen, bei der man mit der geringsten Anzahl an Anweisungen durchkommt. Um Sie in der Kontrolle zu unterstützen, wollen wir zumindest den in der UPN-Logik dargestellten Programmablauf auf einen anderen Grenzwert umstellen.

LBL A	Eingabe des alten Preises
1	
0	
0	Der Rechner setzt den Grenzwert als Vergleichszahl
x⇄y	Der im Y-Speicher befindliche Preis tauscht mit dem in der Anzeige stehenden Grenzwert Platz
x>y	Vergleich Preis – Grenzwert
GTO 1	Ist der eingetastete Preis größer als der Grenzwert, beginnt die Erhöhung um 5% beim Label 1
6	Die Erhöhung um 6% wird hier eingeleitet
GTO 0	Sprung zur Berechnung der Erhöhung und des neuen Preises
LBL 1	Hier beginnt die Erhöhung um 5%
5	
LBL 0	Beginn der Verarbeitung
%	Berechnung der Preiserhöhung
÷	Der neue Preis wird ermittelt
R/S	Das Programm hält mit der Anzeige des neuen Preises an

Wenn wir nun die beiden in der UPN-Logik geschriebenen Programmfolgen miteinander vergleichen, fällt uns einmal auf, daß wir bei der zweiten Variante gleich drei Programmschritte gespart haben. Denn wir haben einerseits einen Grenzwert mit weniger Stellen – 100 anstelle von 99,99 – gewählt und andererseits zweimal je einen Umspeicherungsbefehl eingespart. Dafür benötigen wir einen zustäzlichen Programmschritt für den Speichertausch Grenzwert – Preis. Der gesamte Nutzen beträgt also drei Anweisungen, das sind mehr als 16%. Durch ein wenig Nachdenken und andere Fragestellungen gelingt es uns oft, zu noch besseren Lösungen zu kommen, frei nach dem Motto „Das Bessere ist ein Feind des Guten". In unserem Beispiel haben wir also neben einem **bedingten Sprung** – den nach Label 1 – auch einen **unbedingten Sprung** – den nach Label 0 – nebeneinander verwendet.

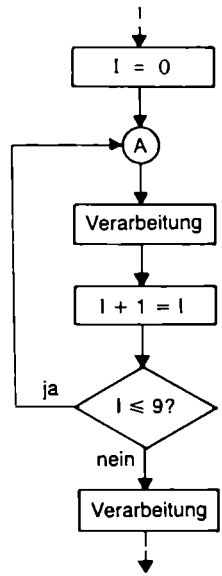
Soweit zu den Verzweigungen, deren Verständnis die größten Schwierigkeiten macht. Lesern, die sich darin noch nicht sicher fühlen, wird empfohlen, das letzte Kapitel zu wiederholen. Je besser man nämlich dieses schwierigste der Kapitel des Programmierens beherrscht, um so leichter fällt dann das Verständnis aller anderen Fragen.

Wir wollen noch einmal kurz zusammenfassen: Ein Sprung bedeutet das Verlassen des normalen Programmablaufes und die Fortsetzung an einer anderen besonders markierten Stelle. Ein Sprung kann ohne jede Vorbedingung, aber auch im Zusammengang mit irgendeiner Frage ausgeführt werden. Je mehr Sprünge in einem Programm eingebaut sind, desto unübersichtlicher wird dieses, bzw. desto mehr Aufmerksamkeit verlangt es vom Programmverfasser. Daher wurde just diesem Abschnitt großer Raum gewidmet, um auch dem in diesen Fragen weniger geschulten Leser ausreichende Möglichkeiten zu bieten, sich mit der Materie vertraut zu machen.

3.4.3. Schleifen

Die Empfehlung, dieses Buch erst dann weiter durchzuarbeiten, wenn man das Problem der Verzweigungen beherrscht, trifft besonders auf dieses Kapitel zu. Denn das Program-

mieren von Schleifen läßt sich direkt aus dem Programmieren von Verzweigungen ableiten. Was versteht man nun unter dem Begriff „Schleife"? Er bezeichnet einen Programmzweig, der mehr als einmal durchlaufen wird und der erst aufgrund einer Bedingung verlassen werden kann. Streng genommen wird nur ein solcher Programmteil als Schleife bezeichnet, der vom Programm aus gesehen mehrmals automatisch durchlaufen wird, ohne daß der Rechner zur Eingabe einer Zahl anhält. Diese Technik benutzen wir in erster Linie bei Näherungsrechnungen, bei denen der Rechner erst bei Erreichen einer vorgegebenen Genauigkeit des Ergebnisses anhält. Ist diese noch nicht im geforderten Ausmaß gegeben, wird der Rechner angewiesen, den zuletzt durchlaufenen Rechenvorgang ein weiteres Mal durchzuführen. Zum besseren Verständnis sei das Wesen einer Schleife auch noch an einem allgemein gehaltenem Beispiel graphisch dargestellt.



Das Grundproblem unseres kleinen Beispieles besteht im zehnmaligen Durchlaufen der nach dem Label A dargestellten Verarbeitung, bevor im Programm weitergegangen werden soll. Die strichlierte Linie zeigt, daß wir uns in einem Programm befinden und uns der Stelle nähern, an der der Wert 0 im Schleifenzähler abgespeichert wird. Beim erstmaligen Eintritt in diesen Programmzweig wird der Label A ignoriert und eine Verarbeitung ausgeführt. Anschließend daran wird der Schleifenzähler um den Wert 1 erhöht. Im Speicher dieses Zählers befindet sich nun der Wert 1. Daraufhin wird gefragt, ob der Inhalt des Zählers I kleiner oder gleich 9 ist. In diesen Fällen – gleich zu setzen mit der Antwort „ja" – wird zum Label A zurückgesprungen und die Verarbeitung in Pfeilrichtung fortgesetzt. Erst wenn der Inhalt des Zählers den Wert 10 erreicht hat, kann die Schleife verlassen und das Programm mit der ersten Verarbeitung nach der Abfrage weitergeführt werden.

Noch eine Bemerkung zur Darstellung der Erhöhung des Speichers um 1. In der Graphik haben wir „I + 1 = I" angeschrieben. Diese unmathematische Gleichung ist im Bereich der

elektronischen Datenverarbeitung durchaus üblich und muß so gelesen werden: Der Wert der Variablen – mathematische Bezeichnung einer allgemeinen Zahl, in unserem Fall eines Datenspeichers I – ist um den Wert 1 zu erhöhen und die Summe im Speicher I abzustellen.

Zu unserem Beispiel zurückkehrend wollen wir vom Programm erfahren, wieviele Preise wir umgerechnet haben, getrennt nach den beiden Preiserhöhungsklassen. Wir möchten aber nicht nur die Gesamtzahl wissen sondern auch, wie sich diese in die mit 5, bzw. 6% umgerechneten Preise aufschlüsselt. Dazu benötigen wir zwei Speicher als Zähler für jede der beiden Erhöhungen. In diese wird nach Durchlaufen des Berechnungsvorganges immer der Wert 1 hineinaddiert.

Nun aber wollen wir diese Programmschleife verlassen und zwar mit der Eingabe des Wertes 0, der ja keiner Erhöhung fähig ist. Wenn der Rechner nun diesen Wert erkennt, soll er nicht den Wert 0 um 5% erhöhen, also letztlich als Ergebnis 0 anzeigen, sondern, an ganz anderer Stelle fortsetzen und die Anzahl der um 5%, dann die um 6% erhöhten Preise und dann die Anzahl der Gesamterhöhungen anzeigen. Diesen Vorgang der Anzeige der Mengen könnten wir sicherlich auch durch einen eigenen Startlabel fixieren; da wir aber Label sparen wollen, machen wir dies mit einem Sprung aus der Schleife.

Obwohl an sich unter einer Schleife ein Programmzweig verstanden wird, der vom Programm aus mehrere Male ohne Eingriff von außen über das Tastenfeld durchlaufen wird, erhält unser Programm durch den Einbau der Abfrage nach dem Wert 0 in der Eingabe durchaus den Charakter einer Schleife, soweit es den mehrmals zu durchlaufenden Teil betrifft. Denn erst bei Erkennen einer derartigen Eingabe wird der als Schleife bezeichnete Programmzweig verlassen, wie dies auch beim automatischen Durchlaufen der Fall ist. Der Wert 0 könnte ja auch in einem anderen Fall vom Programm gesetzt werden.

Zum besseren Verständnis der Zusammenhänge wollen wir auch hier wieder den Programmablauf vorerst einmal verbal darstellen:

1. Eingabe des alten Preises über den Label A.
2. Es erfolgt nun die Abfrage, ob der eingegebene Wert 0 ist.
3. Wenn ja, wird nach dem neu einzurichtenden Label 2 gesprungen und dort die Endausgabe vorbereitet. Dies ist nun der Sprung aus der Schleife!
4. Nun erst setzt der Rechner den Grenzwert 99,99
5. Es erfolgt die Abfrage, ob der Vergleichswert kleiner oder gleich dem alten Preis ist.
6. Im Falle ja, erfolgt der Sprung nach Label 1.
7. Im Falle nein nimmt der Rechner die Zahl 6 in die Anzeige und springt anschließend, nachdem er den Zähler für 6%-ige Erhöhungen um 1 aufgerechnet hat, nach dem Label 0; der Rundenzähler befinde sich im Speicher 6.
8. Der Label 1 wird gesetzt.
9. Der Rechner nimmt hier die Zahl 5 in die Anzeige, und erhöht den Zähler für 5%-ige Erhöhungen um den Wert 1; der Rundenzähler befinde sich im Speicher 5.
10. Der Label 0 wird gesetzt.
11. Der Wert der Erhöhung, bzw. der neue Preis wird berechnet.
12. Der Rechner hält an und zeigt uns den erhöhten Preis an.
13. Hier ist nun der neue Label 2 gesetzt.
14. Der Inhalt des Speichers 5 wird in die Anzeige gerufen, der Rechner hält an.
15. Der Inhalt des Speichers 6 wird in die Anzeige gerufen und der Rechner hält zur Anzeige an.
16. Beide Zahlen werden addiert und das Ergebnis wird angezeigt.

Je nachdem, ob wir den Befehl „Pause" im Tastenfeld unseres PROTARs haben oder nicht, können wir die Ausgabe der Zahlen der umgerechneten Preise unterschiedlich gestalten. Die Anzahlen der 5, bzw. 6%-igen Preiserhöhungen lassen wir uns beispiels-

weise während einer Programmpause anzeigen, erst bei Anzeige des Gesamtergebnisses soll der Rechner anhalten. Ist die PAUSE-Taste nicht vorhanden, müssen wir den R/S-Befehl setzen und die weiteren Anzeigen durch Drücken der R/S-Taste abrufen. Wir haben bei diesem Beispiel zusätzliche Anweisungen einzubauen; **die gestellte Bedingung zum Verlassen der Schleife stellt die Abfrage dar, nach der der eingegebene Wert auf 0 geprüft wird.**

Die Schleife ist in unserem Fall nicht die ideale Schleife, denn das Verlassen dieses immer wieder zu durchlaufenden Programmzweiges wird durch einen äußeren Eingriff provoziert. Nach der Diskussion unseres neu adaptierten Zahlenbeispiels werden wir uns noch ein wenig mit dem Problem einer Schleife beschäftigen und auch die Gefahren zu diskutieren haben, die der Umgang mit Schleifen in sich birgt. Vorerst wollen wir aber die Tastenfolgen unseres Beispiels in beiden Logik-Arten anschreiben:

Tastenfolge in AL

LBL A	Eingabe des alten Preises
STO T	Abspeicherung des Preises im Testspeicher
0	Der Rechner setzt den Wert 0
x = T	Abfrage des eingegebenen Wertes auf 0
GTO 2	Sprung zur Ausgabe der Ergebnisse im Fall 0
9	
9	
9	
9	Der Rechner setzt den Grenzwert
x ≤ T	Vergleich Grenzwert – Preis
GTO 1	Sprung zur Vorbereitung der Erhöhung um 5%, wenn der alte Preis gleich oder größer 100 ist
1	
SUM 6	Der Zählerinhalt für Erhöhungen um 6% wird um 1 angehoben
RCL T	Abruf des eingegebenen Preises aus dem Testspeicher zur Vorbereitung der Erhöhung um 6%
x	Einleitung der Multiplikation
1	
0	
6	Der Rechner setzt den Faktor für 6%-ige Erhöhungen
GTO 0	Sprung zur Auslösung der Multiplikation
LBL 1	Vorbereitung der Erhöhung um 5%
1	
SUM 5	Der Zählerinhalt für Erhöhungen um 5% wird um 1 angehoben

RCL T	Abruf des eingegebenen Preises aus dem Testspeicher zur Vorbereitung der Erhöhung um 5%
x	Einleitung der Multiplikation
1	
0	
5	Der Rechner setzt den Faktor für 5%-ige Erhöhungen
LBL 0	Beendigung der Berechnung der Preiserhöhung . . .
=	. . . durch Drücken der Gleichheitstaste
R/S	Ende der Berechnung der Preiserhöhung und Stop mit Anzeige des neuen Preises
LBL 2	Abruf der Summen der umgerechneten Preise
RCL 5	Abruf der Anzahl der um 5% erhöhten Preise
+	Einleitung der Addition mit der Anzahl der um 6% erhöhten Preise
PAUSE	Anzeige der Anzahl der um 5% erhöhten Preise
RCL 6	Abruf der Anzahl der um 6% erhöhten Preise
PAUSE	Anzeige der Anzahl der um 6% erhöhten Preise
=	Auslösung der Summenbildung
R/S	Ende der Ausgabe der Ergebnisse und Stop mit Anzeige der Anzahl sämtlicher erhöhten Preise

Bei der vorliegenden Programmkonstruktion können wir den Abruf der Ergebnisse, sozusagen um uns den Zwischenstand unserer Arbeit angeben zu lassen, noch vor Beendigung derselben vornehmen. Wollen wir aber eine neue Serie an Preiserhöhungen starten, müssen wir nach dem Gleichheitszeichen und vor dem Programmstop folgende Befehle zum Löschen der Speicher für das Mitzählen der Anzahlen der Erhöhungen einfügen:

0

STO 5

STO 6

Tastenfolge in UPN

LBL A	Eingabe des alten Preises
x = 0	Der Rechner prüft den Wert der Eingabe auf 0
GTO 2	Bei Vorhandensein der 0 wird zur Ausgabe der Ergebnisse gesprungen
9	
9	
.	
9	
9	Der Rechner setzt den Grenzwert
x ≤ y	Vergleich Grenzwert – Preis
GTO 1	Sprung zur Vorbereitung der Erhöhung um 5%, wenn der alte Preis gleich oder größer 100 ist
1	
STO+6	Der Zählerinhalt für Erhöhungen um 6% wird um 1 angehoben
R ↓	Mit der zweimaligen Umspeicherung des Stacks wird der alte Preis, der durch das Setzen des Grenzwertes und der Zahl 1 in den Z-Speicher gewandert ist, wieder in die Anzeige zurückgeholt
R ↓	
6	
GTO 0	Sprung zur gemeinsamen Berechnung
LBL 1	Vorbereitung der Erhöhung um 5%
1	
STO+5	Der Zählerinhalt für Erhöhungen um 5% wird um 1 angehoben
R ↓	Mit der zweimaligen Umspeicherung des Stacks wird der alte Preis aus dem Z-Speicher wieder in die Anzeige zurückgeholt
R ↓	
5	
LBL 0	Beginn der gemeinsamen Berechnung
%	Ermittlung der Preiserhöhung
+	Berechnung des neuen Preises
PRT X	Ausdruck des erhöhten Preises
R/S	Beendigung der Berechnung und Stop bei Anzeige des neuen Preises
LBL 2	Abruf der Summen der umgerechneten Preise
RCL 5	Abruf und . . .
PRT X	. . . Ausdruck der Anzahl der um 5% erhöhten Preise
RCL 6	Abruf und . . .

PRT X	. . . Ausdruck der Anzahl der um 6% erhöhten Preise
+	Bildung der Gesamtsumme der erhöhten Preise und . . .
PRT X	Ausdruck dieser Zahl
R/S	Ende der Ausgabe der Ergenisse und Stop mit Anzeige der Gesamtsumme der Anzahl der erhöhten Preise.

Auch hier hätten wir die Löschung der Speicher 5 und 6 mit der Befehlsfolge O STO 5 STO 6 vor dem Programmstop vornehmen können.

Diese zwei mal zwei Umspeicherbefehle können vermieden werden, wenn wir nach dem Label A den Befehl zur Zwischenspeicherung des eingegebenen Preises in den Speicher 0 mit der Anweisung **STO 0** einfügen. Anstelle von den je zwei Umspeicherungsbefehlen setze ich die Anweisung **RCL 0**, mit dessen Hilfe ich den eingegebenen Preis wieder in die Anzeige zurückhole. Bei dieser Vorgangsweise kann ein Programmspeicherplatz eingespart werden.

Bei den verschiedenen Demonstrationen unseres Übungsbeispiels wird bewußt zwischen der Multiplikation und der Prozentmethode gewechselt. In der Praxis wird man jene wählen, für die sich der vorhandene PROTAR am besten eignet.

Wir haben hier stillschweigend die dritte Möglichkeit der Ausgabe der Ergebnisse vorgenommen, indem wir die PAUSE-Anweisungen durch eine Druckanweisung ersetzt haben. Im Falle des in UPN dargestellten Beispiels sind wir auch wieder von der ursprünglichen Form ausgegangen, bevor wir im vorigen Abschnitt durch Umänderung in der Fragestellung einen Vorteil durch Einsparung von Programmspeichern erzielt haben. Auch hier ist wieder die gleiche Möglichkeit gegeben, durch Änderung der gestellten Bedingung zu versuchen, wertvolle Programmspeicherplätze einzusparen. Wir haben auch vor Ende der Berechnung eines neuen Preises einen Druckbefehl eingebaut, um die erhöhten Preise auch ihrem Werte nach zu dokumentieren. Die allmähliche Weiterentwicklung unseres anfangs so einfachen Beispieles zeigt, welchen Nutzen man aus einem PROTAR ziehen kann. Man muß sich nicht auf die Errechnung der am dringendsten benötigten Werte beschränken, sondern kann sich durch einen entsprechenden Ausbau des Programmes vielfältige zusätzliche Informationen sichern.

An dieser Stelle wollen wir uns aber eines sparen, nämlich das Abtesten mit Zahlenbeispielen. Die eingefügten Programmergänzungen sind so klar, daß es dieser Kontrolle nicht mehr bedarf.

Programmschleifen werden in ihrer idealen Form in jenen Fällen eingesetzt, in der wir zum Beispiel Iterationsrechnungen durchführen wollen. Solche Näherungsrechnungen werden solange durchlaufen, bis das Ergebnis mit der gewünschten Genauigkeit vorliegt. Diese Genauigkeit können wir vorgeben, indem wir sagen, bis zu welcher Dezimalstelle Gleichheit bestehen muß, bzw. ab welcher Unterschiede toleriert werden. Hier haben wir es mit einer **Schleife** zu tun, bei der vom Rechner bestimmt wird, wann diese Schleife verlassen werden darf; eben dann, wenn der berechnete Wert mit der geforderten Genauigkeit vorliegt.

Der Ablauf einer derartigen Berechnung kann stark vereinfacht so dargestellt werden:

- Label A zur Eingabe und zum Start.
- Rechenvorgang nach einer bestimmten Formel durchführen.
- Abspeicherung Ergebnis.
- Vergleich mit dem Ergebnis des vorletzten Rechenganges.
- Bei Gleichheit Stop.
- Bei Nichtgleichheit Veränderung der Vorgabe und zurück zu Label A.

Nach dem Rücksprung wird der Rechenvorgang erneut aufgenommen. Im Falle der Gleichheit auch der verlangten Genauigkeit könnte anstelle des Programmstops selbstredend ein Sprungbefehl eingebaut sein, der den Rechner anweist, mit dem erzielten Ergebnis in einem anderen Rechenzweig fortzufahren. Bei solchen Näherungsrechnungen kommt es oftmals zu mehreren Durchläufen, bis sich das gewünschte Ergebnis einstellt. Hier zeigt sich ganz besonders der Vorteil von PROTAR. Dieses stark vereinfacht dargestellte Problem bei einer Iterationsrechnung ist sozusagen das **klassische Beispiel einer Programmschleife.**

Es wurde schon mehrfach auf die Notwendigkeit einer genauen Einschätzung der Bedingungen hingewiesen. Speziell bei der idealen Form einer Schleife finden wir diese Forderung nur allzu berechtigt. Wir müssen hier nämlich trachten, daß die Bedingung, die wir stellen, auch erfüllbar ist. Die gestellte Forderung sollte speziell im Fall einer Iterationsrechnung in absehbarer Rechenzeit realisiert werden können. Stelle ich eine übertriebene Genauigkeit, kann es passieren, daß sie überhaupt nicht erfüllbar wird und der Rechner sich damit in einer **ewigen Schleife** befindet. In diesem Falle, wenn die Berechnung unverhältnismäßig lange gedauert hat, können wir fast schon mit Sicherheit rechnen, daß wir uns in einer ewigen Schleife selbst gefangen haben. Hier hilft nichts mehr anderes, als den Rechner mit dem Drücken der Start/Stop-Taste aus der Verlegenheit zu helfen, in die wir ihn aus unserer Schuld hineinmanövriert haben. Die Ursache dieses Umstandes kann auch von wo anders her kommen; es muß nicht nur die Bedingung die Ursache sein, es kann auch die Vorgabe, die Starteingabe falsch gewählt worden sein; das hängt von der Art der Iterationsrechnung ab.

Daher kontrollieren wir einige Rechenvorgänge und versuchen die Veränderungen des durch die Iterationsrechnung ermittelten Wertes zu erforschen und setzen danach unsere Maßnahmen, entweder verändern wir die Eingabe oder stellen eine weniger harte Bedingung. Diese Kontrolle können wir mit dem bereits besprochenen SST-Knopf durchführen. Wir rechnen einige Beispiele durch, indem wir das Programm schrittweise durch Betätigung der SST-Taste „abfahren". Über die Anzeige können wir die Zwischenergebnisse leicht kontrollieren und stellen relativ rasch fest, wo die Fehlerursachen liegen. Denn leider zeigt sich ein Fehler niemals am Ort der Entstehung, sondern an ganz anderer Stelle im Programm.

Manche PROTARe haben für diese Kontrollrechnung auch eine weitere Einrichtung: In einer bestimmten Schalterstellung – sie wird TRACE genannt – zeigt uns der Rechner Schritt für Schritt an, was er gerade macht, indem immer Programmbefehl, den er ausführt und Zwischenergebnis am Papierstreifen ausgedruckt werden. Das erleichtert jedes Fehlersuchen sehr! Allerdings müssen wir uns dann durch einen längeren Kontrollstreifen durcharbeiten!

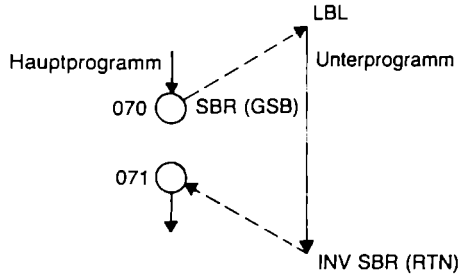
Die Darstellung dieser verhältnismäßig schwer verständlichen Probleme wurde bewußt nicht an einem bestimmten Rechner veranschaulicht. Damit soll der firmenneutrale Charakter des Buches erhalten bleiben. Andererseits wird dem Leser geholfen, die Probleme auch „abstrakt" sehen zu lernen und die Übersicht trotz immer härter werdender Anforderungen nicht zu verlieren. Diese Übung wird Sie bei der Erstellung eigener Programme und bei der Diskussion der Beispiele im Anhang befähigen, rasch die jeweils dargestellte Problematik zu erfassen.

3.5. Unterprogramme

Im letzten Kapitel über Elemente der Programmierung wollen wir uns mit einer weiteren Besonderheit der Programmgestaltung befassen.

Ein Unterprogramm ist ein Programmteil, der, wie schon sein Name ausdrückt, dem Hauptprogramm untergeordnet ist. Er wird vom Hauptprogramm „aufgerufen" und nach

Abarbeitung des Unterprogramms wird die Arbeit wieder im Hauptprogramm fortgesetzt. Im Hauptprogramm wird ein dem Sprungbefehl ähnlicher Befehl gesetzt. Anstelle der GTO-Taste wird die SBR, bzw. GSB-Taste gedrückt. Der genannten Taste muß wiederum eine Sprungadresse folgen. Dieser angesprungene Label leitet das Unterprogramm ein. Am Ende dieses Programmteiles darf allerdings nicht die R/S-Anweisung gesetzt werden, weil dann die Rückkehr in das Hauptprogramm verhindert wäre. Je nach Rechner sind die Tasten INV und SBR, bzw. die RTN-Taste zu drücken. Während eines normalen Sprunges, ausgelöst durch das Setzen des GTO-Befehles an einer bestimmten Programmstelle, gibt es keine Rückkehr an jene Stelle im Programm, von wo aus der Sprung erfolgt ist. Bei den erwähnten Tasten allerdings, mit deren Hilfe ein Unterprogramm „angesprungen" wird, „merkt" sich der Rechner jene Stelle, von wo aus er das Hauptprogramm verlassen hat und kehrt bei der Rückkehr ins Hauptprogramm an die jeweils darauffolgende Stelle zurück. Bei der SBR-, bzw. GSB-Anweisung setzt sich der Rechner also eine **Rücksprungadresse**, z. B. die Zeilennummer 070 in sein „Rücksprunggedächtnis" und kehrt, nachdem er im Unterprogramm die Rücksprunganweisung mit der Tastenfolge INV SBR, bzw. RTN gefunden hat, an die Stelle 071 ins Hauptprogramm zurück.



Das Unterprogramm wird dabei meist an eine Stelle nach dem Hauptprogramm gesetzt, wo es am wenigsten stört. Unterprogramme plaziert man im allgemeinen zusammen an das Ende des ganzen Programmes, auch um die Übersicht leichter bewahren zu können, als wenn diese an Stellen, an den sie zwar auch nicht stören, aber doch im Wege sind, irgendwo mitten im Hauptprogramm ihren Platz hätten.

Diese Anordnung, die allein im Interesse der besseren Durchschaubarkeit der Programmgestaltung getroffen wurde, sollte solange beibehalten werden, als sich der Leser noch nicht in vollem Maße in der Programmierung sattelfest weiß. Erst im Verlaufe der weiteren Praxis zeigt sich, daß eine übertriebene Übersichtlichkeit Rechenzeit kosten kann.

Man wird daher Unterprogramme so anordnen, daß sie vom Hauptprogramm betrachtet, die kürzeste Suchzeit benötigen. Die Lösung dieser Frage hängt von der Art des Rechners ab, in welcher Form der Programmspeicher nach dem Label des Unterprogrammes abgesucht wird. Auf diese Umstände wurde bereits im Abschnitt 3.3. über Programm-Marken hingewiesen. Bezüglich des Einflusses der schon mehrfach erwähnten Rechenzeit wird auf den Abschnitt 4.7.6. verwiesen.

Warum Unterprogramme?

Eine berechtigte Frage! Mitunter werden Programmteile zu Unterprogrammen umfunktioniert, weil sie mehrfach in der gleichen Reihenfolge im Programm auftauchen und daher mehr Programmspeicherplatz benötigen. An sich wären Unterprogramme nicht erforderlich, wenn das Programm in einer Form vorliegt, in der es ohnehin nicht den ganzen Programmspeicherplatz des Rechners benötigt. Dabei ist zu beachten, daß das Aufsuchen von Unterprogrammen im allgemeinen mehr Rechenzeit in Anspruch nimmt, als wenn

diese zwar gleichartigen Programmstellen doch jede für sich blieben. Also Unterprogramme zur Einsparung von Programmspeicherstellen nur dann, wenn Mangel an derartigen Speicherstellen herrscht.

Wir haben bereits an unserem Übungsbeispiel gesehen, wie sich ein Programm ausweiten kann, wenn man genügend Anforderungen an das Programm stellt! Daher sei gleich an dieser Stelle die **Wirtschaftlichkeit von Unterprogrammen** näher untersucht.

Zunächst eine Faustformel, um diese sofort ermitteln zu können:

$$n \geqslant \frac{2 + \Sigma B}{\Sigma B - 1}$$

Dabei bedeuten

ΣB jene Anzahl von Befehlen, die sich mehrmals wiederholen (sprich Summe B).

n bedeutet darin die Anzahl an Wiederholungen, ab der ein Einsatz eines Unterprogramms vertretbar erscheint.

Im Zähler dieser Formel steht die Zahl 2, die zur Summe der sich wiederholenden Anweisungen addiert werden muß. Warum? Deswegen, weil ja ein Unterprogramm zusätzlich zwei Befehle benötigt, nämlich den Label und die Anzeige des Endes. Hier ist allerdings eine Einschränkung zu machen für jene Rechner, für die die LBL-Anweisung und Label-Adresse in zwei Programmzeilen zu schreiben ist; für diesen Fall muß anstelle der Zahl 2 die Zahl 3 im Zähler eingesetzt werden. Und warum muß im Nenner von der Summe B der Wert 1 abgezogen werden? Nun weil an jeder Stelle im Hauptprogramm, von der aus das Unterprogramm benötigt wird, eine Sprunganweisung einzusetzen ist.

Beispiele

ΣB	=	2	3	4	5	6	7	8	9	10
n	=	4	2,5	2	1,75	1,60	1,50	1,43	1,38	1,33

Man kann sich daraus auch eine noch einfachere Faustformel ableiten und sagen, daß ab vier sich wiederholenden Befehlen ein zweimaliges Anspringen eines Unterprogrammes bereits wirtschaftlich ist. Bei drei sich wiederholenden Anweisungen müssen es bereits drei Sprünge sein und bei zwei sich wiederholenden Befehlen ist ein viermaliges Anspringen eines Unterprogramms gerade schon vertretbar; in diesem Falle gibt es allerdings noch keine Einsparung an Programmspeicherplätzen.

Ein weiterer Grund, ein Unterprogramm einzurichten, kann aus **organisatorischen Gründen** statt aus Einsparungsgründen gegeben sein. Wir haben beispielsweise eine Abfrage gestellt, bei der im Falle der Antwort „ja" ein Programmzweig zusätzlich zu durchlaufen wäre, der aber im Falle der Antwort nein ausgespart bleiben müßte. Nun wird dieser als Unterprogramm dargestellt, das mit dem der Abfrage folgenden Befehl aufgesucht wird. Nach Rückkehr ins Hauptprogramm erfolgt die Fortsetzung im gleichen Programmteil, in dem der Rechner auch bei einer Antwort „nein" seine Arbeit weitergeführt hätte.

Des weiteren kann es auch vorkommen, daß Teile eines Hauptprogramms durch geschickte Anordnung gleichzeitig auch als Unterprogramm fungieren. Man muß nur immer darauf achten, daß jeder als Unterprogramm fungierende Programmzweig mit einem **Label beginnt** und mit einem **INV SBR** oder **RTN abgeschlossen** sein muß. Hält man sich diese Tatsache stets vor Augen, kann einem nichts passieren.

Zur Veranschaulichung soll wieder unser Zahlenbeispiel mit den Preiserhöhungen herangezogen werden. Wir wollen die zuletzt erarbeitete Version weiterverwenden, in der wir das Problem einer Schleife erörtert hatten. Aus Platzgründen werden wir allerdings hier im Buch

nicht mehr die gesamte Programmfolge anschreiben, sondern nur jene Teile, in denen wir etwas ändern wollen. Dementsprechend ist beim Studium der folgenden Programmieranweisungen auf den Abschnitt 3.4.3. zurückzublättern.

Bei der in **AL festgelegten Tastenfolge** kommen die beiden Befehle RCL T und das Multiplikationszeichen zweimal hintereinander vor. Wir wissen allerdings, daß diese geringe Häufigkeit einen wirtschaftlichen Einsatz eines Unterprogrammes noch nicht rechtfertigt; übungshalber setzen wir aber diese beiden Anweisungen in ein Unterprogramm und plazieren es ganz an den Anfang.

LBL 3	Beginn des Unterprogrammes
RCL T	Abruf des alten Preises zur Vorbereitung auf die Multiplikation mit einem %-Satz
x	
INV SBR	Ende des Unterprogrammes

Anstelle dieser beiden Programmschritte RCLT und x müssen wir nun im Hauptprogramm jeweils die Anweisung

SBR 3

setzen.

Mit dem Label 3 eröffnen wir die Existenz unseres Unterprogramms und mit INV SBR weisen wir den Rechner an, ins Hauptprogramm zurückzukehren und beim Setzen der Prozentzahl 6, bzw. 5 fortzusetzen.

Das gleiche wiederholen wir bei der in **UPN dargestellten Tastenfolge.** Hier finden wir allerdings nur die beiden hintereinandergesetzten Umspeicherbefehle R ↓, die wir in ein Unterprogramm verpacken könnten. Dieses setzen wir gleich nach dem Hauptprogramm.

LBL 3

R ↓

R ↓

RTN

An den Stellen dieser beiden Umspeicherbefehle im Hauptprogramm müssen wir nun hier den Befehl

GSB 3

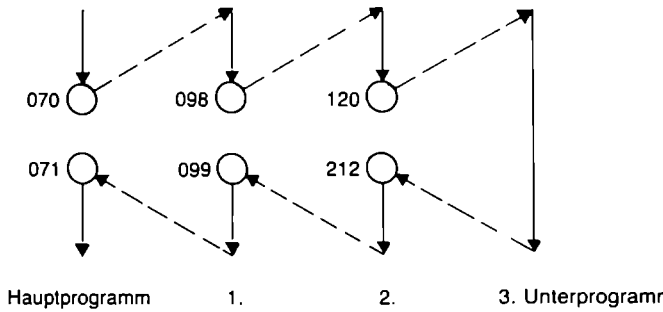
setzen.

Eines merken wir wieder, was wir an anderer Stelle diskutiert haben, der Label 3 ist der vierte verwendete Label, weil wir beim Label 0 zu zählen begannen.

Es kam hier darauf an, die Verwendung eines Unterprogramms zu demonstrieren und nicht darauf, eine Wirtschaftlichkeit nachzuweisen. Wir wissen, daß wir in diesem Fall in der Praxis selbstverständlich kein Unterprogramm installiert hätten, da wir die zwei Anweisungen umfassende Tastenfolge nur zweimal im ursprünglichen Programm vorliegen hatten. Die Wirtschaftlichkeit von Unterprogrammen hängt einerseits ab von der **Anzahl der eingesparten Programmspeicherplätze** und andererseits – vor allem bei rechenintensiven Programmen – von der **Rechenzeit**. Man wird daher nur dann Unterprogramme verwenden, wenn es die geringe Anzahl der zur Verfügung stehenden Programm-Speicherplätze gebietet. Wir wollen uns auch diese Regel zu eigen machen.

Innerhalb eines Unterprogrammes kann übrigens ein weiteres Unterprogramm zur Hilfe herangezogen werden. Verschiedene Rechner können dies mehrmals tun; in diesem Fall hat der Rechner die Fähigkeit, alle Rücksprungadressen zu speichern. Dagegen können Rechner der mittleren Datentechnik, vor allem für kommerzielle Anwendungen, oft nur zwei sogenannte **Unterporgrammebenen** realisieren, weil eben im Falle eines derartigen Einsatzes von den Problemstellungen nicht mehr gefordert wird.

Im Falle des Vorliegens von drei Unterprogrammebenen könnte dies wie folgt bildlich dargestellt werden.



Zusammenfassung: Folgende Momente sind bei der Erstellung eines Programmes in dieser Reihenfolge zu berücksichtigen:

1. Die Übersichtlichkeit sollte vor allem vom Anfänger allen anderen Punkten vorangestellt werden. Er muß sein Programm auch noch zu einem späteren Zeitpunkt im Ablauf richtig verfolgen können. Denn je länger, bzw. häufiger man ein Programm benutzt, um so mehr taucht der Wunsch nach zusätzlichen Informationen auf, die oftmals durch eine nur geringfügige Programmerweiterung gewonnen werden können. Der Einbau dieser zusätzlichen Anweisungen kann nur bei vollständiger Kenntnis der alten Programmabläufe richtig vorgenommen werden.
2. Sparen an Programmspeicherplätzen stellt ebenfalls eine Forderung an die optimale Gestaltung von Programmen dar.
3. Rechenzeiten sollten vor allem bei rechenintensiven Programmen berücksichtigt werden. Doch darüber lesen Sie mehr im Abschnitt 4.7.6.

4. Das Programm

Bisher wurden die Elemente der Programmierung im Zusammenhang mit kleinen Programmbeispielen dargestellt. Nun aber zum Programm in seiner Gesamtheit. Die programmtechnischen Probleme treten nun etwas in den Hintergrund. Dieser Abschnitt befaßt sich somit mit dem Programm im weitesten Sinne der Bedeutung, also vorrangig mit der Organisation von Programmen.

Wir wissen, daß **ein Programm nichts anderes ist, als eine mehr oder minder sinnvolle Aneinanderreihung von Anweisungen an den Rechner zur Lösung eines bestimmten Problems**. Mit einem Programm, das sämtliche denkbaren Lösungsansätze enthält, wird der Rechenvorgang gesteuert und kontrolliert. Insofern hat das Programm auch eine **Kontroll- und Steuerfunktion,** im besonderen im Hinblick auf die hinlänglich diskutierten Programmverzweigungen.

Der sarkastische Hinweis auf weniger sinnvolle Anweisungen in der Einleitung zu diesem Kapitel kann wie folgt ausgelegt werden: Ein Rechner ist nur „mäßig" intelligent. Er vermag abzuprüfen, ob eine Marke auch gesetzt ist, die Sie im Ablauf eines Programmes aufrufen, oder ob Sie sonst irgendwelche formalen Fehler begangen haben. Er reagiert dann „sauer", indem er auf bereits geschilderte Weise zu blinken beginnt oder sein ERROR in der Anzeige aufleuchten läßt. PROTAR vermag aber nicht zu beurteilen, ob Sie nicht doch 55 meinten, als sie irrtümlicherweise die Zahl 65 eingetastet haben.

Sie können sich aber selbst gegen Fehleingaben absichern, indem Sie im Rahmen des zur Verfügung stehenden Programmspeicherangebotes von Eingaben kontrollierenden Abfragen Gebrauch machen, z. B. zur Verhinderung der Eingabe von negativen Werten oder Dezimalzahlen. Für diese Fälle „basteln" Sie einprägsame **Fehlerausgänge;** so. z. B. weisen Sie den Rechner an, im Falle, daß er eine von Ihnen vorgegebene Eingabe als falsch erkennt, an eine bestimmte Programmstelle zu springen, um dort beispielsweise 9999999999 99 anzuzeigen. Sie wählen diese Fehlerausgabe so, daß sie selbst die Anzeige nicht als Ergebnis, sondern als Fehlerausgabe interpretieren.

Des weiteren wird man sich beispielsweise bei Kalkulationen Positions- oder Schlüsselzahlen anzeigen lassen, um immer sicher zu gehen, was man gerade eingegeben hat oder was man als nächstes eingeben muß. Auch diese Dinge gehören zu einem optimalen Programm, einerseits um den Programmablauf richtig steuern zu können, andererseits um auch von der Bedienung her immer eindeutig den Stand des Ablaufes klarzustellen. Speziell die letzte Forderung wird im kommerziell-betriebswirtschaftlichen Bereich häufig zu stellen sein. Technische Probleme werden im allgemeinen ohne wesentliche Unterbrechungen abgewickelt, was für den kaufmännischen Bereich wohl nicht immer gilt.

Und gerade an solchen Programmschnittpunkten, wo das Programm zur Aufnahme neuer Eingaben anhalten muß, wird es zwangsläufig zu Fehlern kommen und um diese soweit als nur irgend möglich einzudämmen, sind die genannten Kontrollen unbedingt vorzusehen.

Die Beantwortung der Frage nach dem Begriff „Programm" war bisher doch eher unvollständig, denn wir haben mehr Mühe auf die Skizzierung des Endproduktes gelegt, das uns seine Dienste auf Anforderung – sprich Tastendruck – zur Verfügung zu stellen hat und das auch in allerdings schematischer „Denkungsweise" tut, somit so gut, wie der Mensch es vorgegeben hat. Sie sehen damit wohl recht deutlich, daß zu einem gut funktionierenden Programm doch etwas mehr gehört, als nur einige Befehle oder Anweisungen an den Rechner hinzuschreiben. Ein gutes „Drum-Herum", also eine gute Organisation um dieses Programm ist ebenso wichtig, wie das Programm selbst. Das gilt nicht nur für unsere PROTAR betreffenden Programme, sondern ist grundsätzlich auf Programme für alle Rechenanlagen zu beziehen.

Meist haben wir es mit Problemen zu tun, die relativ komplex sein können und Schwierigkeiten bieten beim Versuch, eine Problemlösung in einen so kleinen Computer hineinzustopfen, wie es eben einmal unser PROTAR ist.

Mit ein wenig Übersicht ist es in manchen Fällen doch möglich, wenn man auch in einigen Punkten Abstriche machen muß. Es war so z. B. möglich, in einen Rechner mit 224 Programmschritten und 26 Datenspeichern eine Anlagenbuchhaltung hineinzustopfen, die entweder manuell oder automatisch ablaufen kann, die die kalkulatorischen Kosten, wie Abschreibungen und Zinsen, entweder in Jahres- oder Monatsbeträgen errechnen kann, die ferner vier Anlagengruppen zu unterscheiden vermag und manches andere mehr. Überdies sammelt sie auch noch die Gesamtsummen dieser kalkulatorischen Kosten getrennt nach den einzelnen Aufwandarten und die Summe der Wiederbeschaffungswerte. Dieses Programm bot außerdem Gelegenheit, das Problem so aufzubereiten, daß es als Vorlage für ein Programm für einen Bürocomputer dienen konnte. Im übrigen werden in den Beispielen D und E im Anhang zum Problem der Anlagenbuchhaltung viele Hinweise gegeben. Wir wollen aber nun daran gehen, ein Programm gemeinsam zu erstellen und bedienen uns dabei unseres bereits mehrfach veränderten Beispieles.

Ein Programm muß nicht unbedingt eine einzige Folge von zusammenhängenden Anweisungen sein. Wie wir gehört haben, können mehrere Programmzweige zusammen ein Programm bilden. Mit den sogenannten „freien Funktionstasten", das sind die Marken mit den Buchstaben A, B, C, D und E, sowie A', B', C' D' und E' bzw. a, b, c, d und e können verschiedene Programmteile markiert werden und so als selbständige Zweige in einem Programm fungieren.

Meist sind es die zehn genannten Labeln, die auf einem PROTAR als „freie Funktionstasten" eingerichtet sind. Dabei können die Tasten mit den Buchstaben A, B, C, D und E als **Primär-Labeln** bezeichnet werden, weil sie direkt, also ohne Vorwahltaste, verwendet werden, während die restlichen – A' bis E' in einer Modellversion, bzw. a bis e in einer anderen – als **Sekundär-Labeln** bezeichnet werden, eben weil sie nur in Verbindung mit einer Vorwahltaste aktiviert werden können. Es können aber auch die **numerischen Label** – meist von 0 bis 9 – direkt über das Tastenfeld angewählt werden. Dazu ist die SBR-, bzw. GSB-Taste und die Label-Adresse zu drücken und die Sache funktioniert in gleicher Weise wie bei den **Alpha-Labeln**. Durch diese Vielzahl an Möglichkeiten ist der individuellen Gestaltung Tür und Tor geöffnet, sodaß trotz der geringen Gerätegröße der PROTARe die dadurch bedingten engen Grenzen doch so erweitert werden können, daß der Benutzer eines PROTARes erst einmal für seinen eigenen Bedarf die Möglichkeiten auszuloten hat, bis er an Grenzen stößt, die dann trotz findigster Programmgestaltung nicht überschritten werden können. Aber bis zur Erreichung dieser Grenze hat er noch ein weites Feld der Nutzung vor sich. Die Auslotung aller Eigenschaften wird erst durch die fortgeschrittene Praxis ermöglicht.

Ein Blick in den Anhang wird Sie sicherlich davon überzeugen, daß sich mit PROTAR eine Vielzahl von Problemen bewältigen läßt. Ohne den Wert von Standardprogrammen mindern zu wollen, ist festzustellen, daß speziell im Bereich der kommerziellen Datenverarbeitung ein individuell auf die eigenen Bedürfnisse zugeschnittenes Programm wirtschaftlicher ist. Der Aufwand für die Durcharbeitung des Buches und für die spätere Programmierung lohnt sich dann, wenn eine programmiert lösbare Aufgabe zu Zeiteinsparungen gegenüber einer konventionellen Lösungsmethode führt. Dies ist auf jeden Fall dann gegeben, wenn eine Aufgabenstellung häufig auftaucht.

4.1. Problemanalyse

Aufgrund des zentralen Charakters und der besonderen Wichtigkeit der Problemanalyse wird diesem Abschnitt breiter Raum gewidmet. Je besser ein Problem durchleuchtet wird, desto besser wird dann das daraus abgeleitete Programm funktionieren.

Bevor wir also in die Lage versetzt werden, ein Programm zu schreiben und es dem Rechner einzugeben, müssen wir uns über die Zielsetzung, die wir mit unserem Programm verfolgen, im klaren sein.

Hinsichtlich der Organisation rund um das Programm müssen wir klarstellen, welche Eingaben benötigt werden und woher diese zu nehmen sind. Der letzte Hinweis ist deswegen erforderlich, um allenfalls während der Datenverarbeitung auftretende Rückfragen positiv erledigen zu können. Des weiteren ist festzuhalten, in welcher Form die Ergebnisse vom Rechner ausgegeben werden müssen und was mit ihnen zu geschehen hat, bzw. wem dieselben weiterzugeben sind.

Außerdem ist der Ablauf der Rechenvorgänge selbst zu analysieren und der Verarbeitung auf PROTAR anzupassen.

Die drei wesentlichsten Punkte der Problemanalyse kurz zusammengefaßt sind

- festzuhalten was unser Problem darstellt und wie es gelagert ist,
- wie wir es lösen wollen und
- welcher Hilfsmittel wir uns dabei zu bedienen haben.

Was ist unser Problem?

Wir müssen lernen, unser Problem so mit Worten zu umreißen, daß es nicht nur für uns für spätere Überarbeitungen des Programmes leicht verständlich ist. Unter Umständen muß damit gerechnet werden, daß auch einer unserer Mitarbeiter mit dem Programm zu arbeiten hat. Denn aus der Problemanalyse werden die wichtigsten Punkte für die Programmbeschreibung und Bedienungsanleitung abgeleitet.

Im Falle unseres Übungsbeispieles sollte folgendes erreicht werden:

1. Preise unter 100,00 sollen um 6%, Preise mit 100,00 und darüber dagegen nur um 5% erhöht werden.
2. Es soll festgestellt werden, wieviele Preise um 5% und wieviele um 6% erhöht werden; außerdem ist auch die Gesamtzahl der Erhöhungen festzustellen.

Das Problem ist somit klar formuliert.

Wie wollen wir unser Problem lösen?

Unser einfaches Übungsbeispiel eignet sich besonders gut als Demonstrationsobjekt, weil die folgenden Darlegungen infolge der Einfachheit des Beispieles leicht faßlich werden. Auf diese Weise wird dem Leser die Lösung auch komplizierterer Vorgänge wesentlich erleichtert.

Zunächst ermitteln wir jene Rechenformeln, die uns die einfachste Lösung unseres Rechenproblems ermöglichen.
Bezüglich des Rechenvorganges hatten wir zwei Varianten angenommen, je nachdem, ob unser PROTAR eine %-Taste besitzt oder nicht. Hat er sie, läuft der Rechenvorgang wie folgt ab:

Alter Preis	90,00
6% Erhöhung	5,40
Neuer Preis	95.40

oder

Alter Preis	120,00
5% Erhöhung	6,00
Neuer Preis	126,00

Rechner, die diese Taste nicht besitzen, können das Problem auf eine andere Art lösen:

$$P_n = P_a \times (1 + p/100);$$

dabei bedeuten

P_n . . . neuer Preis
P_a . . . alter Preis
p . . . Prozentsatz der Erhöhung.

Wenn Sie nun die vorhin gewählten Werte einsetzen, werden Sie zu den gleichen Ergebnissen kommen. Wir errechnen also den neuen Preis in Form einer Multiplikation; den Formelteil 1 + p/100 dürften wir indes von einer bekannten kaufmännischen Formel, der Rentenformel her kennen, in der für den erwähnten Formelteil die Bezeichnung q gewählt wird. Bei dieser Rechenmethode erhält man direkt den neuen Preis durch Multipliaktion des alten Preises mit dem Aufschlagsfaktor q. Der Rechner zeigt dann allerdings nicht die absolute Preiserhöhung pro Position an. Wer auch diese Größe angezeigt haben möchte, muß einen Rechner mit %-Taste wählen.

Mit einem erhöhten Aufwand an Rechenoperationen läßt sich allerdings dieses Problem so lösen, daß auch bei einem Rechner ohne %-Taste die absolute Preiserhöhung angezeigt wird.

Dazu müssen wir den alten Preis unbedingt in einem Datenspeicher abstellen. Dann berechnen wir die Preiserhöhung allein, indem wir den alten Preis P_a mit dem durch 100 geteilten Prozentsatz der Erhöhung, also mit p/100, multiplizieren. Dieses Zwischenergebnis lassen wir uns dann anzeigen und addieren den Betrag in den Speicher hinein, in dem wir den alten Preis zwischengespeichert haben. Zur Ausgabe des neuen Preises müssen wir die so errechnete Summe aus diesem Zwischenspeicher in die Anzeige holen. Aufgrund des höheren Aufwandes wird diese Variante aber nicht weiter verfolgt.

Nun haben wir auch die Forderung zur Feststellung der Zahl der umgerechneten Preise gestellt. Zu diesem Zweck müssen wir also zwei Speicher als Zählwerke einrichten, in die wir jeweils vor oder nach Durchlauf eines Rechenvorganges den Wert 1 hineinaddieren und zwar in jenen, der dem gerade verwendeten Erhöhungsprozentsatz zugeordnet ist. Wir wählen als einprägsame Speicher-Nummer, da ja sonst kein Speicher benötigt wird, außer allenfals den Speicher mit der Adresse 0 bzw. T, für die Zwischenspeicherung des alten Preises zur Feststellung der Anzahl der mit 5% erhöhten Preise den Speicher 5 und für die Anzahl der um 6% erhöhten Preise den Speicher 6. Damit scheint auch dieser Punkt für unser Beispiel im großen und ganzen klar umrissen. Wenn wir uns nun bei einer manuellen Bearbeitung unserer Arbeit genau und selbstkritisch beobachten, wird es uns nicht schwer fallen, die Arbeitsvorgänge im Ablauf zu erfassen. Dieses „Sich-selbst-auf-die Finger-Schauen" macht uns am Anfang sehr zu schaffen. Mit einer verbalen Beschreibung lernen wir unser Problem auch transparenter darzustellen. Dabei stoßen wir mitunter auf Details, deren Berücksichtigung bisher in der konventionellen Datenverarbeitung so automatisch erfolgte, daß wir bei der Lösung der Aufgaben mit Hilfe eines Programmes beinahe darauf vergessen hätten.

Wir sehen, daß wir unsere Probleme erst auf die Verwendung eines bestimmten Rechners zuschneiden müssen; meist genügen dazu kleine Veränderungen. Man möchte meinen, daß das überall so sein könnte. Im Bereich unserer großen Brüder in der DV könnte aber folgende Geschichte passiert sein:

Der Chef einer großen Firma ruft eines Tages seine Abteilungsleiter und Führungskräfte zu einer Besprechung zusammen. Dabei teilt er ihnen mit, daß er gehört habe, die DV sei eine große und wunderbare Sache, die er auch haben möchte. So ganz sicher sei er sich seiner Sache aber nicht und beauftragt einige seiner Leute, sich dieser Sache anzunehmen und gibt ihnen eine Frist von etwa einem Jahr, um das Problem der Einführung der DV als neue Organisationsform zu studieren und vorzubereiten. Nach dieser Frist sollten diese Leute Bericht erstatten und auch gleich eine Wirtschaftlichkeitsrechnung vorlegen.

Nun werden Unterlagen studiert, Vorträge und Seminare besucht, Firmen eingeladen, Vorschläge zu unterbreiten und vieles andere. Das wesentliche, was diese Leute aber anstellen oder anstellen sollten, ist das, daß sie in Ruhe von Abteilung zu Abteilung gehen, um dort die Arbeitsabläufe zu studieren. Sie versuchen auch dabei, die Mengen an Daten zu ergründen und erkundigen sich in Einzelgesprächen über die Arbeitsaufgaben jedes einzelnen. Bei diesen Gelegenheiten erkennen sie auch schon Stolpersteine, die sie gleich mit erledigen, d. h. sie führen Änderungen durch, sie verbessern die Abläufe, verschieben allenfalls auch Kompetenzen, arbeiten neue Formblätter aus, kurz, sie greifen da und dort in das Geschehen ein, denn sie wissen, daß verschiedene Dinge sonst nicht auf eine DV in der vorliegenden Form übernommen werden können.

Nach Ablauf des Jahres und Vorliegen der Ergebnisse dieser Aufnahme erstatten sie Bericht: Man habe festgestellt, daß sich nach dem letzten Stand der Dinge kaum Chancen für den Einsatz einer elektronischen Datenverarbeitungsanlage ergeben, denn man habe gleichzeitig mit den Vorarbeiten gewisse notwendige Rationalisierungsmaßnahmen setzen müssen, die schon einen so guten Rationalisierungseffekt ergeben haben, daß sich die Installation einer DV im Augenblick nicht zu rentieren scheine. Es sind aber da und dort doch einige Stellen, die als Engpässe anzusprechen sind, wo eine raschere Verarbeitung vorteilhaft erschiene; aber die Probleme sind dort so klein, wenn auch in einer Vielzahl vorkommend, daß sich deswegen die Anschaffung einer größeren Rechenanlage auch nicht rentieren dürfte. Man schlägt dem Chef für diese Abteilungen die Anschaffung einiger PROTARe vor, was von ihm auch sofort genehmigt wird, weil die Investitionskosten, hierfür relativ gering sind.

Ein paar Jahre sind vergangen, immer mehr Mitarbeiter sind mit PROTARen ausgerüstet worden, bis sich da und dort eine unüberwindbare Grenze auftut. Wiederum werden Überlegungen angestellt, der Betrieb ist auch mit den PROTARen gut durchorganisiert worden und nun kann man die Wirtschaftlichkeit einer größeren Rechenanlage aufgrund des immer größer werdenden Anteiles der Personalkosten an den Gesamtkosten nachweisen. Außerdem hat man immer mehr **analytisch denken gelernt** und hat den Zeitpunkt zur Installation der DV gut gewählt. Die PROTARe wurden dann aber nicht in die Ecke gestellt, sondern erfüllen nach wie vor ihre Aufgabe in dem für sie zutreffenden Rahmen.

Eine unwahre Geschichte, aber möglich wäre sie. Meinen Sie nicht auch?

Damit sollte nur dargestellt werden, wie wichtig eine **Analyse** der anstehenden Probleme ist, und daß am Ende einer Untersuchung nicht immer eine große Rechenanlage stehen muß, PROTARe tun es mitunter auch, bzw. können eine wichtige Ergänzung einer großen Datenverarbeitungsanlage darstellen.

Aber nun zurück zu unserem Beispiel: Stellen Sie sich vor, Sie hätten um die 1.600 Preise umzurechnen und müßten das auf konventionelle Weise erledigen. Das heißt, Sie müßten 1.600 mal den Prozentbetrag eintasten, 1.600 mal die Prozenttaste und 1.600 mal die Funktionstaste drücken. Durch Verwendung eines Programms für diese Aufgabe ersparen Sie sich dagegen 1.600 mal 2 Tastenoperationen, das sind 3.200 Tastendrücke. Rechnen Sie etwa 2 bis 3 Tastendrücke pro Sekunde, kommen Sie auf eine Zeitersparnis von 15 bis 20 Minuten bei der Bewältigung einer einzigen Aufgabe. Berücksichtigt man dabei noch, daß es sich um die Einsparung **nicht-kreativer** Arbeit handelt und daß man gleichzeitig

3.200 möglichen Eingabefehlern entgeht, läßt sich ausrechnen, wie schnell sich ein PROTAR und die für die Programmierung erforderliche Zeit amortisiert.

Es wurde festgestellt, daß es im Grunde zwei Lösungsmöglichkeiten gibt, aus der wir eine im Einklang mit der Ausstattung unseres PROTARes auswählen. Dabei ist zu erwähnen, daß die Variante, den alten Preis mit dem Aufwertungsfaktor q zu multiplizieren, auf allen Rechnermodellen anwendbar ist.

Welcher Hilfsmittel haben wir uns zu bedienen?

Die Frage nach den Hilfsmitteln brauchte bei unserem Übungsbeispiel wegen dessen Einfachheit an sich erst gar nicht gestellt zu werden. Aber auch hier gilt wiederum der Grundsatz, daß Überlegungen in allen jenen Fällen anzustellen sind, bei denen die Probleme nicht so einfach und übersichtlich gelagert sind. Dazu kommt noch, daß man oft Unterlagen in einer nicht direkt verwertbaren Form vorliegen hat und diese erst in eine auf PROTAR verwertbare Darstellung umformen muß. Hier kann man die Preise nach den alten Preislisten eingeben und die neuen Preise neben die alten schreiben. Diese handschriftlich geänderten Listen bilden dann die Vorlage für einen Neudruck.

Sicherlich haben alle diese Bemerkungen wenig, wenn überhaupt, etwas mit den Vorgängen im Rechner zu tun. **Aber die Arbeit mit jeder Art von Computer ist so gut, wie es die ihn umgebende Organisation ist.** Und das gilt ebenso ohne Abstriche auch für den Umgang mit PROTAR. Möglicherweise sind in dem Fall, wo wir daran gehen, mit „ihm" ein Problem lösen zu wollen, auch Formblätter neu zu entwickeln, gewisse Arbeiten anders zu verteilen, kurz die Organisation auf die neue Rechenhilfe abzustellen, dabei auch die Gedanken zu entwickeln, wie das Ziel, das wir erreichen wollen, am besten vorzubereiten ist.

Das wesentliche daran ist die **Organisation;** PROTAR stellt nur eines der möglichen und ausgezeichneten Hilfsmittel dar, dieser Organisation zu dienen. Die Behandlung des Problems der Durchleuchtung unserer Aufgaben ist mir ein besonderes Anliegen; denn gerade damit, wenn Sie auch an Grenzprobleme denken, werden auch Sie in die Lage versetzt, in Hinkunft selbständig Problemlösungen zu erarbeiten und Programme zu erstellen.

4.2. Ablaufschema

Wenn wir geschickt vorgegangen sind, haben wir schon in der Problemanalyse unsere Aufgabe so formuliert, daß es uns leicht fällt, das Ablaufschema direkt zu zeichnen.

Was ist nun ein Ablaufschema? Die Antwort liegt bereits in dem durch diesen Begriff gegebenen Inhalt. Es ist ein Schema, graphisch dargestellt, in dem der gesamte Ablauf eines Vorganges beschrieben wird.

Eine graphische Darstellung hat den Vorteil einer im allgemeinen besseren Übersichtlichkeit gegenüber einer verbalen Beschreibung, wenn auch auf eine solche nicht immer zu verzichten ist! Aus der graphischen Darstellung erkennen wir leichter die Verarbeitungsrichtung vor allem im Falle der vielzitierten Verzweigungen und in allen Fällen, in denen im Programm auch einmal rückwärts gesprungen wird. Sicherlich wird man in der Praxis einen Ablaufplan nur für solche Aufgaben zeichnen, die nicht so klar liegen, wie dies in unserem Übungsbeispiel der Fall ist.

Mit dieser Art von Darstellung erkennen wir leichter, wo wir allenfals etwas vergessen haben könnten und können diese „missing links" – die fehlenden Gelenke – noch rechtzeitig einsetzen, ohne erst anläßlich eines Programmtests darüber zu stolpern.

Anläßlich der Erstellung des Ablaufschemas müssen wir auch überlegen, welche Speicher wir benötigen, wo und wann eventuell Zwischenergebnisse abgespeichert werden müssen. Im Falle unseres Beispiels haben wir die Speicher 5 und 6 zum Abstellen der Zähler für die Ermittlung der Anzahlen der um 5% bzw. 6% zu erhöhenden Preise festgelegt. Ein gemeinsamer Zähler ist nicht erforderlich, da für die Ausgabe, bzw. Ermittlung der Gesamtzahl der erhöhten Preise die Inhalte der Speicher 5 und 6 addiert werden können. Müssen wir Preise oder den Wert der Erhöhung zwischenspeichern? Eigentlich nein, außer mit UPN bei Verzicht auf die Umspeicherungsanweisungen zugunsten der Zwischenspeicherung des alten Preises im Speicher 0. Doch halt, wenn wir Rechner mit AL benutzen, müssen wir den **Testspeicher** zum Abstellen des Preises aktivieren, da wir ja diesen Wert, bzw. die Eingabe einmal auf 0, das andere Mal auf kleiner 100 abfragen müssen. Bei mit der UPN arbeitenden PROTARen ist dies nicht unbedingt erforderlich, da wir ja mit den vier Arbeitsspeichern ausgerüstet sind und dort umspeichern können.

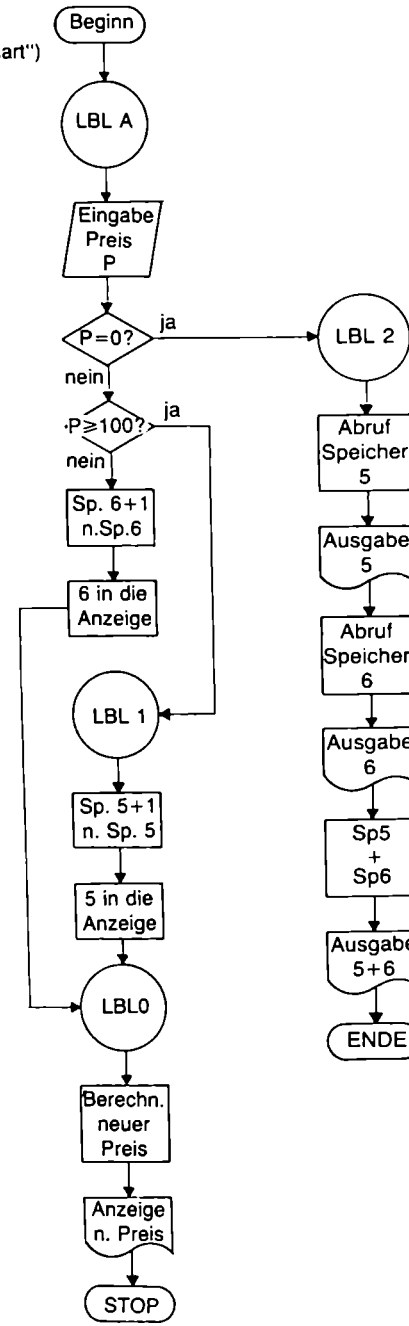
Ursprünglich hatten wir uns für zwei Eingabe-Labeln entschlossen, d. h. wir wollten die mit 5, bzw. 6% zu erhöhenden Preise über zwei verschiedene Tasten, A und B, eingeben. Wir sind davon aber wieder abgerückt. Zumal für den Abruf der Ergenisse eine dritte Taste hätte gewählt werden müssen. Wir können aber noch einen anderen Grund angeben, warum wir uns auf eine Taste beschränken wollen: Auf diese Weise vermeiden wir es, durch Fehlbedienung einen Preis mit einem falschen Prozentsatz zu erhöhen. Am neuen Preis merken wir den Irrtum kaum, denn zwischen 5 und 6% ist nicht soviel Unterschied, daß er uns im Betrag oder im neuen Preis auffallen würde. Damit, daß wir es dem Rechner überlassen, diese Entscheidung zu treffen, sind wir eine Sorge los. Und wie wir uns bereits entschieden haben, lösen wir mit der Eingabe der Zahl 0 den Ergebnisabruf aus. Die Wahl der 0 ist neutral, denn einen Preis mit dem Wert 0 gibt es nicht und kann mit einem noch so hohen Prozentsatz auch nicht erhöht werden. Daher benützen wir diesen Umstand zum Zweck des Abrufes der Ergebnisse und haben somit sämtliche Funktionen einer einzigen Taste zugeordnet. Das Drücken einer anderen Alpha-Taste ist ohne Bedeutung, da der Rechner einen „nicht definierten" Label, also einen, den wir in unserem Programm nicht verwenden, nicht „kennt" und daher bei der Anwahl eines solchen Labels nach einer Weile des Suchens zu blinken beginnt bzw. das Wort ERROR in der Anzeige erscheinen läßt. Also sind wir auch hier wieder gegen Fehleingaben weitestgehend abgesichert; gegen einen falsch eingetasteten Preis ist PROTAR allerdings machtlos, denn er führt die Berechnung auftragsgemäß auch mit einer falschen Eingabe durch. Eine solche kann aber nur zu einem falschen Ergebnis führen. Der Vollständigkeit halber wollen wir nun die verbale Formulierung auf den letzten Stand bringen.

So lautet nun die detaillierte Ablaufbeschreibung unseres Übungsbeispiels:

1. Eingabe eines alten Preises, gleich welcher Höhe, über den Label A, bzw. der Zahl 0 zum Abruf der Ergebnisse.
2. Der Rechner fragt nun ab, ob der eingegebene Wert 0 ist.
3. Im Falle der Antwort ja, wird zum Label 2 gesprungen, von wo aus die Ausgabe der Anzahlen der umgerechneten Preise erfolgt. Mit der Zahl 0, die kein Preis ist, steuern wir die Ausgabe der Ergebnisse.
4. Wird die Frage verneint, setzt der Rechner den Grenzwert 100.
5. In diesem Stadium wird nun gefragt, ob der eingegebene Wert größer oder gleich 100 ist.
6. Im Falle ja, erfolgt der Sprung nach Label 1, von wo aus der Rechner die Preiserhöhung um 5% vorbereitet.
7. Im Falle der Verneinung der Frage wird vorerst der Inhalt des Zählers für 6%-ige Preiserhöhungen um 1 erhöht.
8. Der Rechner setzt die Zahl 6 in die Anzeige.
9. Das Programm springt nun nach Label 0 und beginnt dort die Berechnung des um 6% zu erhöhenden Preises.

Das Ablaufschema

(Ablaufdiagramm, englisch „flow chart")



10. Der Label 1 wird gesetzt, denn hier wird die Preiserhöhung um 5% vorbereitet.
11. Der Inhalt des Zählers für 5%-ige Preiserhöhungen wird um 1 erhöht.
12. Der Rechner setzt die Zahl 5 in die Anzeige.
13. Der Label 0 wird gesetzt, von wo aus nun die gemeinsame Berechnung des neuen Preises beginnt.
14. Der Wert der Erhöhung, bzw. der neue erhöhte Preis wird berechnet.
15. Der Rechner hält an und zeigt uns den neuen Preis an. Vor dem Anhalten kann auch ein Druckbefehl eingesetzt werden, um das Ergebnis zu dokumentieren.
16. Der Label 2 wird gesetzt; hier beginnt die Ausgabe der Anzahlen der umgerechneten Preise.
17. Der Inhalt des Speichers 5 wird in die Anzeige gerufen.
18. Der Rechner hält kurz zur Anzeige der Anzahl der mit 5% umgerechneten Preise an und setzt dann von selbst fort, wenn die Pause-Taste im Programm-Modus gesetzt wurde. Bei einem Stop anstelle einer Pause ist zur Anzeige der übrigen Ergebnisse das Programm mit dem Drücken der R/S-Taste wieder ingang zu setzen. Auch hier könnte anstelle der Anzeige en Druckbefehl gesetzt werden.
19. Der Inhalt des Speichers 6 wird in die Anzeige gerufen.
20. Der Rechner hält kurz zur Anzeige der Anzahl der mit 6% erhöhten Preise an und setzt dann wieder von selbst fort. Hier gilt die in Punkt 18 gemachte Zusatzbemerkung in gleicher Weise. Auch hier wäre wieder ein Druckbefehl denkbar.
21. Die Inhalte der beiden Speicher 5 und 6 werden zusammengezählt.
22. Die Summe wird als Gesamtzahl der erhöhten Preise angezeigt. Folgerichtig wäre auch hier eine Druckanweisung zu setzen.
23. Der Rechner hält an; das Programm ist beendet.

Diese Darstellung nimmt auf die Unterschiedlichkeit der Rechner nicht Bedacht und berücksichtigt auch nicht die jeweiligen Unterprogramme.

Auf einen bestimmten Rechner bezogen, läßt sich ein solches Ablaufschema siehe Seite 88, natürlich noch verfeinern. Darauf kann sich dann die Codierung, das ist die Programmdarstellung in Tastenschritten, beziehen.

4.3. Festlegen der Programmschritte

Jede Erörterung einer Programmiertechnik muß sich letzten Endes in der Niederschrift eines Programmes in Form von Anweisungen oder Befehlen an den Rechner dokumentieren. Wenn wir also Analyse und Ablaufschema beherrschen, können wir an das Umsetzen unserer Problemlösung in Programmschritte gehen. Diesen Vorgang bezeichnet man mit dem Fachausdruck **Codierung**.

Das Festlegen der Programmschritte ist mitunter nicht nur abhängig von der jeweils verwendeten Rechenlogik, sondern auch von einzelnen Modellen innerhalb einer Modellreihe. So kann es beispielsweise Unterschiede in der Formulierung von Label und Labeladresse geben Diese beiden Tastendrücke können zwei oder auch nur eine Programmzeile belegen. Über derartige Einzelheiten geben uns aber wiederum die Bedienungsanleitungen Auskunft. Daher werden wir auch bei der Angabe der Tastenschritte allfällig nötige Vorwahltasten nicht eingehen. Trotzdem dürfte es bis auf die Berücksichtigung einiger rechnerabhängiger untergeordneter Details möglich sein, unser bereits in Programmform vorliegendes Übungsbeispiel sofort auszuprobieren.

Bevor wir jedoch die Erörterung unseres Übungsbeispieles fortsetzen, wollen wir hier noch einige Bemerkungen zur Kapazität unseres PROTARs machen, die zur Aufnahme der einzelnen Programmschritte zur Verfügung steht.

Bei längeren Programmen kann es bei der Festlegung der Programmschritte eine Überschreitung der Programmspeicherkapazität geben. Wir kommen erst gar nicht einmal zur Eingabe und zum Test eines Programmes. Daher müssen wir die erst in Abschnitt 4.4.5. zu diskutierende Redigierung eines Programms vorziehen. Es gilt, Programmspeicherplatz frei zu machen.

Mitunter gelingt dies, wenn man das Programm ein wenig umorganisiert. Dabei werden Rechenabläufe kontrolliert, ob diese nicht mit weniger Programmschritten und damit etwas weniger umständlich abgewickelt werden können. Multiplikationen mit 100.000 zum Beispiel werden wir mit den Anweisungsfolgen 1 EE 5, bzw. EEX 5 ausführen. Man wird nachsehen, ob sich gewisse Folgen von Anweisungen in der genau gleichen Art wiederholen und aus diesem Grund in ein Unterprogramm – siehe Abschnitt 3.5 – packen lassen. Anläßlich der Suche nach zusätzlichen Speicherplätzen werden mitunter wirkungslose Befehle und Befehlsfolgen – auch solche gelangen manches Mal in ein Programm! – eliminiert. Wirkungslos sind einerseits Anweisungen, die vom Programm aus nie erreicht werden, oder Befehlsfolgen, die wie die Addition mit der Zahl 0 ohne Auswirkungen auf die Ergebnisse sind. Es gibt hier viele Möglichkeiten; bei entsprechender Praxis werden Sie manche weitere, hier nicht erwähnte Einsparungsvariante entdecken!

4.3.1. Beispiel in der algebraischen Logik

Die im Abschnitt 3.5. Unterprogramme dargestellte Programmversion verändern wir nur in einem Punkt. In unserem dort in AL formulierten Übungsbeispiel wählen wir die Multiplikationsmethode anstelle der Prozentrechnung. Allerdings haben wir dort das Beispiel nicht komplett angeschrieben, um die Bedeutung des verwendeten Unterprogrammes herauszustreichen. Im übrigen bleiben wir dabei, das Unterprogramm vor das Hauptprogramm zu setzen, weil mit AL ausgestattete PROTARe auf diese Weise das Unterprogramm im allgemeinen schneller finden.

LBL 3	Unterprogramm zum Abruf des alten Preises und Vorbereitung zur Multiplikation mit Aufwertungsfaktor q
RCL T	Abruf des alten Preises aus dem Testspeicher in die Anzeige
x	Einleitung der Multiplikation mit q
INV SBR	Ende des Unterprogramms und Rückkehr ins Hauptprogramm
LBL A	Eingabe-Label für alte Preise
STO T	Zwischenspeicherung des alten Preises im Testspeicher
0	Der Rechner setzt die Zahl 0 in die Anzeige
x = T	Abfrage der eingegebenen Zahl auf den Wert 0
GTO 2	Im Fall der Antwort ja wird zur Ausgabe der Anzahlen der umgerechneten Preise gesprungen
9	
9	
9	
9	Der Rechner setzt den Grenzwert in die Anzeige
x ≤ T	Abfrage der Größenordnung der Eingabe

GTO 1	Ist die eingegebene Zahl größer als der Grenzwert, wird zur Erhöhung um 5% gesprungen.
1	Hier beginnt die Erhöhung um 6%
SUM 6	Der Zählerinhalt für 6%-ige Erhöhungen wird um den Wert 1 angehoben
SBR 3	Im Unterprogramm wird der alte Preis abgerufen und die Multiplikation mit dem Setzen des ersten Operanden und dem Funktionszeichen vorbereitet
1	
.	
0	
6	Der Rechner setzt den zweiten Operanden $q_6 = 1 \div \frac{6}{100}$
GTO 0	Sprung zum Label 0 zur Berechnung des neuen Preises
LBL 1	Hier beginnt die Erhöhung um 5%
1	
SUM 5	Der Zählerinhalt für 5%-ige Erhöhungen wird um den Wert 1 angehoben
SBR 3	Im Unterprogramm wird der alte Preis abgerufen und die Multiplikation mit dem Setzen des ersten Operanden und dem Funktionszeichen vorbereitet
1	
0	
5	Der Rechner setzt den zweiten Operanden $q_5 = 1 \div \frac{5}{100}$
LBL 0	
R/S	Ende der Berechnung und Anzeige des neuen Preises
LBL 2	Beginn der Ausgabe der Anzahlen der umgerechneten Preise
RCL 5	Abruf der Anzahl der um 5% erhöhten Preise
+	Setzen des Funktionszeichens zur Summenbildung der Gesamtzahl der umgerechneten Preise
PAUSE	Anzeige der Anzahl der um 5% erhöhten Preise während einer Programmpause
RCL 6	Abruf der Anzahl der um 6% erhöhten Preise
PAUSE	Anzeige der Anzahl der um 6% erhöhten Preise während einer Programmpause
=	Summenbildung beider Anzahlen
R/S	Stop und Anzeige der Gesamtzahl der erhöhten Preise

4.3.2. Beispiel in der umgekehrten polnischen Notation

Im Vergleich zur im Kapitel 3.5. Unterprogramme dargestellten Version setzen wir dieses Beispiel ohne Änderungen an, nur daß wir es hier komplett bringen. Im erwähnten Kapitel hatten wir nur das Unterprogramm zur Umspeicherung des Stack-Speichers behandelt. Wiederum nicht berücksichtigt haben wir die Programmspeicher sparende Darstellung der Abfrage. Unberücksichtigt bleibt auch die Variante zur Vermeidung der zweimal vorkommenden beiden Umspeicherbefehle, die wir im Abschnitt 3.4.3. Schleifen bei der Diskussion des in UPN dargestellten Übungsbeispiels erörtert haben. Hier halten wir an der Umspeicherung mit Hilfe des Unterprogrammes fest.

LBL A	Eingabe-Label für alte Preise
x = 0	Abfrage der eingegebenen Zahl auf den Wert 0
GTO 2	Im Fall der Antwort ja wird zur Ausgabe der Anzahlen der umgerechneten Preise gesprungen
9	
9	
9	
9	Der Rechner setzt den Grenzwert in die Anzeige
x ≤ y	Abfrage der Größenordnung der Eingabe
GTO 1	Ist die eingegebene Zahl größer als der Grenzwert, wird zur Erhöhung um 5% gesprungen
1	Hier beginnt die Erhöhung um 6%
STO+6	Der Zählerinhalt für 6%-ige Erhöhungen wird um den Wert 1 angehoben
GSB 3	Umspeicherung des Stacks im Unterprogramm, um den alten Preis, der durch das Setzen des Grenzwertes und der Zahl 1 in den Z-Speicher gewandert ist, wieder in die Anzeige zu bringen
6	Der Rechner setzt die Prozentzahl 6 in die Anzeige
GTO 0	Sprung zum Label 0 zur Berechnung der Preiserhöhung
LBL 1	Hier beginnt die Erhöhung um 5%
1	
STO+5	Der Zählerinhalt für 5%-ige Erhöhungen wird um den Wert 1 angehoben
GSB 3	Umspeicherung des Stacks im Unterprogramm, um den alten Preis wieder in die Anzeige zu bringen, der durch das Setzen des Grenzwertes und der Zahl 1 in den Z-Speicher gewandert ist
5	Der Rechner setzt die Prozentzahl 5 in die Anzeige
LBL 0	Beginn der Berechnung der Preiserhöhung
%	Ermittlung der absoluten Erhöhung
+	Ermittlung des neuen Preises

PRT X	Ausdruck des erhöhten Preises
R/S	Stop und Ende der Berechnung; in der Anzeige erscheint der neue Preis ebenfalls.
LBL 2	Beginn der Ausgabe der Anzahlen der umgerechneten Preise
RCL 5	Abruf der Anzahl der um 5% erhöhten Preise und ...
PRT X	... Ausdruck dieser Zahl
RCL 6	Abruf der Anzahl der um 6% erhöhten Preise und ...
PRT X	... Ausdruck dieser Zahl
+	Addition beider Anzahlen und ...
PRT X	... Ausdruck deren Summe
R/S	Stop unter gleichzeitiger Anzeige dieser Summe
LBL 3	Unterprogramm zur Umspeicherung des Stacks, um den alten Preis, der durch das Setzen des Grenzwertes und der Zahl 1 inzwischen in den Z-Speicher gewandert ist
R ↓	
R ↓	
RTN	Ende des Unterprogrammes und Rückkehr ins Hauptprogramm

Das Unterprogramm hätte noch besser nach dem Ende der Berechnung und vor dem Label 2 eingeschoben werden können. Die Suchzeit vom Hauptprogramm aus gesehen wäre kürzer gewesen.

4.4. Programmverarbeitung

Nach der Erstellung der Grundlagen zur Programmierung unseres Übungsbeispieles gehen wir nun an die praktische Durchführung des Projektes.

4.4.1. Programmeingabe

Vor der Eingabe des codierten Programmes in den Rechner ist in der Bedienungsanleitung nachzusehen, wie man den Rechner zur Aufnahme des Programmes einstellt. In der Fachsprache ausgedrückt, heißt das, wie wir PROTAR in den **Programm-Modus** versetzen müssen. Dies geschieht nach dem Einschalten des Rechners entweder durch Betätigung eines besonderen Schalters oder einer bestimmten Taste; in beiden Fällen erscheinen zwei oder drei Nullen in der Anzeige. Sie bedeuten, daß die Programmzeile mit der Zeilennummer 0 angezeigt wird. Jede im Programm-Modus gedrückte Taste bewirkt keine Rechenoperation, sondern eine Belegung eines Programmspeichers mit einer Anweisung.

Die Anzeige besteht aus zwei Zifferngruppen:

Entweder wird	35	06
oder	035	06

angezeigt. Die erste Gruppe, aus zwei oder drei Ziffern bestehend, zeigt die **Zeilennummer** – hier 35 – an. Die besagt entweder, welche Zeile als nächste einzutasten ist – in diesem Fall besteht die zweite Gruppe aus lauter Nullen – oder welche Zeile als letzte eingetastet wurde.

Die zweite Zifferngruppe bezeichnet dann den jeweiligen **Tasten-Code**. Dieser Code besteht aus bis zu drei zweistelligen Ziffernuntergruppen, wobei jede Untergruppe die Position einer gedrückten Taste anzeigt. Zuvor wurde in der Zeile 35 die Zahl 6 eingetastet. Eine Untergruppe 42 besagt, daß die gedrückte Taste in der vierten Reihe an zweiter Stelle zu finden ist. Hinsichtlich verschiedener Tasten-Codes gibt es bei einzelnen Rechnern Besonderheiten, über die die jeweilige Bedienungsanleitung hinreichend aufklärt.

In manchen Fällen stehen in dieser zweiten Zahlengruppe nur Nullen. Das kommt daher, daß dieser Rechner, wie zuvor erwähnt, jene Zeile angibt, in die der nächste Programmbefehl aufzunehmen ist. Soll der eingegebene Befehl kontrolliert werden, muß die BST-Taste – „back-step-Taste" – gedrückt werden. Vor der Eingabe der nächsten Anweisung darf in dem Fall nicht vergessen werden, den ursprünglichen Zustand mit Hilfe der SST-Taste – „single-step-Taste" – wieder herzustellen.

Nun geben wir Schritt für Schritt der codierten Befehlsfolge dem Rechner ein. Alle **Codier-Blätter**, das sind Formulare, in die die Anweisungen eingetragen werden, haben neben der Formulierung des Befehls eine eigene Spalte, in die die Tasten-Codes einzutragen sind. Hier muß ich für meinen Teil eingestehen, daß diese Spalte bei mir meist frei bleibt. Denn im Bedarfsfall sieht man in der Tabelle der Code-Bezeichnungen einfach nach und die meisten Codes kann man selbst aus der Tastenlage feststellen, mit Ausnahme einiger Sonderfälle, die aber alle in den Bedienungsanleitungen genau beschrieben sind.

Wir haben nun Schritt für Schritt unserer Tastenfolge dem Rechner eingegeben und müssen, um das Programm nutzen zu können, den Rechner in den ursprünglichen Zustand, den Rechen- oder Run-Modus, zurückversetzen. Das geschieht dadurch, daß man den Schalter auf die ursprüngliche Position zurückstellt oder jene Taste, mit der man den Programm-Modus angewählt hat, ein zweites Mal drückt. Diese Taste heißt bezeichnenderweise „Learn"-Taste! Damit weiß nun PROTAR, daß wir rechnen wollen.

4.4.2. Test

Nun kommt wohl der spannendste aller Augenblicke, denn wir wollen untersuchen, ob sich unser PROTAR auch alles gemerkt hat, was wir ihm eingegeben haben, bzw. ob die Eingabe auch ohne logische Fehler erfolgt ist. Zu diesem Zweck bereiten wir uns einige Rechenbeispiele vor, die wir aber vorher im Kopf, bzw. mit Papier und Bleistift, lösen. Damit läßt sich ein Test rasch und effektvoll abwickeln. Ein kleiner Abklatsch eines Prinzips, das wir noch von der Schulzeit her kennen und zwar vor allem aus dem Mathematik-Unterricht: Wir denken uns das Problem gelöst und rollen den Lösungsablauf sozusagen von hinten auf. Wenn wir ein Programm testen, sollten wir immer solche Beispiele wählen, die konventionell gerechnet leicht nachvollziehbar sind. So bekommen wir langsam nicht nur Vertrauen zu PROTAR, sondern auch zu uns selbst und unseren Fähigkeiten, funktionstüchtige Programme zu schreiben. Für jeden eingebauten Rechenzweig sollte man ein paar einfache Beispiele entwickeln und diese unter Umständen mehr als einmal abtesten.

Der Test ist ein wesentliches Element der Programmierung, weniger was den Grad der Schwierigkeit betrifft, als den der Sicherheit, daß das Programm in der Praxis auch funktioniert, wenn wir es nicht mehr so leicht haben sollten, die Richtigkeit eines Ergebnisses abzuschätzen. Jedes Programm sollte, bevor es seinen Dienst in der Praxis antritt, zuvor auf Herz und Nieren getestet werden.

Es braucht auch nicht weiter betont zu werden, daß – zurückkommend auf den Test unseres Beispiels – in jeder der beiden Logik-Arten, **Ausgangspunkt und Ergebnis gleich sein müßten**. Die Betonung liegt auf dem Wort „müßten", denn ein Fehler ist nie auszuschließen, vor allem wenn wir in der Hitze des Gefechtes womöglich einfach nur vergessen haben sollten, einen Befehl einzutasten. Aber das sollte uns bei unserem ersten Programm und bei einem relativ einfachen Beispiel wohl kaum passieren!

Wir bereiten uns nun einige Beispiele vor:

Alter Preis		Erhöhung %		Wert		Erhöhung		Neuer Preis
20,00	+	6%	=	20,00	+	1,20	=	21,20
100,00	+	5%	=	100,00	+	5,00	=	105,00
30,00	+	6%	=	30,00	+	1,80	=	31,80
200,00	+	5%	=	200,00	+	10,00	=	210,00
120,00	+	5%	=	120,00	+	6,00	=	126,00
40,00	+	6%	=	40,00	+	2,40	=	42,40
160,00	+	5%	=	160,00	+	8,00	=	168,00

Dabei erkennen wir, daß wir vier Preise um 5% und drei Preise um 6% erhöht haben. Diese Anzahlen benötigen wir ja auch bei Abschluß der Berechnung der verschiedenen Erhöhungen. Insgesamt waren es also sieben Erhöhungen.

Nun vergewissern wir uns nochmals, ob wir den Rechner nicht nur eingeschaltet, sondern ihn nach Eingabe des Programms wieder in den Run-Modus zurückgestellt haben.

Hätten wir ihn in der Zwischenzeit ausgeschaltet und hätten das Programm nicht auf einen Magnetstreifen übernommen, unter der Annahme, einen Rechner mit einer Schreib-/Leseeinrichtung vor uns zu haben, müßten wir es wie im Abschnitt 4.4.1. beschrieben, neu eintasten. Nachdem wir uns überzeugt haben, daß alles in Ordnung ist, geben wir den ersten Preis – in unserem Fall 20,00 – ein. Dazu tasten wir die Zahlen 2 und 0 hintereinander ein und drücken anschließend die Taste A. Nachdem wir das Übungsbeispiel für beide Logikarten programmiert haben, wollen wir als erstes die AL-Version überprüfen. Der praktische Ablauf ist aber in beiden Fällen der gleiche; beide müssen zu den gleichen Ergebnissen führen. Nachdem wir also die Taste A gedrückt haben und nachdenklich auf die flimmernde Anzeige blicken, hält der Rechner endlich und zeigt uns den Wert 1,06! Das kann doch nicht der um 6% erhöhte erste eingegebene Preis sein? Hat uns unser PROTAR angeschmiert! Denken wir; aber wir versuchen es ein zweites Mal, wieder das gleiche. Wir probieren es mit dem zweiten Preis und geben 100,00 ein. Ergebnis: Der Rechner zeigt zur Abwechslung die Zahl 1,05 an. Nein soviel Schadenfreude ist PROTAR doch nicht zuzutrauen und wir werden doch uns gegenüber mißtrauisch. Nun überprüfen wir im Programm-Modus unsere Eingaben mit der SST-Taste, nachdem wir vor der Umschaltung mit der RST-Taste den Programmzeiger an den Anfang gestellt haben. Mit jeder Betätigung der SST-Taste wird der nächstfolgende Befehl angezeigt. Zuvor haben wir im Run-Modus ein Rechenbeispiel kontrolliert und es für richtig befunden. Die Anzeige der Zahlen 1,06, bzw. 1,05 zeigen uns – wenn wir in unserer Tastenfolge nachsehen – das gleiche Bild: Der Rechner setzt den Aufwertungsfaktor, denn um den handelt es sich bei der Anzeige, kommt dann zum Label 0 und damit zur Berechnung der Erhöhung und bleibt dann mit der Anzeige des neuen Preises stehen. Stehen bleibt er ja, aber den neuen Preis zeigt er uns nicht an, sondern offensichtlich den Aufwertungsfaktor! Uns schwant etwas; möglicherweise haben wir vergessen, die Gleichheitstaste zu setzen! In AL ist ja die Gleichheitstaste zu drücken, um einen Rechenvorgang vollständig abzuschließen.

Wir sehen uns nochmals Schritt für Schritt an, indem wir immer wieder die SST-Taste drücken und die Anzeige verfolgen. Und nun stellen wir tatsächlich fest, daß wir mit unserer Vermutung recht hatten: Nach Eintasten der letzten Ziffer des Aufwertungsfaktors – in dem Fall 5 oder 6 – haben wir den Label 0 und darauf gleich die R/S-Taste gesetzt; da also lag der Hase im Pfeffer! Wir ergänzen nun die fehlende Anweisung, indem wir uns vorher in der Bedienungsanleitung vergewissert haben, wie das anzustellen ist, schalten wieder in den Run-Modus zurück, löschen sämtliche Inhalte der Datenspeicher – wir haben ja bei den ersten Versuchen die Zähler trotz falscher Ergebnisse benutzt! – und geben nun nochmals

20,00 ein und erhalten nun die richtige Antwort von PROTAR: 21,20. So geben wir nun die weiteren sechs Beispiele ein und erhalten nun immer das richtige Ergebnis. Zum Abschluß geben wir zum Ergebnisabruf über den Label A den Wert 0 ein. Wir hatten ja vorgesehen, daß uns dieser Wert zur Ausgabe der Anzahlen der umgerechneten Preise führen soll. Dies geschieht auch, denn zuerst erscheint die Zahl 4 in der Anzeige, nach etwa einer Sekunde die Zahl 3 und nach einer weiteren Sekunde die Zahl 7, worauf der Rechner stehen bleibt. Vier Preise wurden um 5%, drei um 6% erhöht und insgesamt wurden sieben Preise umgerechnet.

Übungshalber hatten wir unser Beispiel auch in der UPN-Version programmiert. Auch diese wird dem Test unterzogen. Vorsichtshalber sehen wir vorher nach, ob wir nicht auch hier Unterlassungssünden begangen haben und gehen das Programm Schritt für Schritt durch. Dabei erkennen wir keinerlei Fehler und gehen an den Test, der uns tatsächlich die gleichen Ergebnisse bringt, wie wir sie schon zuvor erhalten haben; eine Bestätigung, daß das Programm in der UPN-Version von vorne herein fehlerfrei gearbeitet hat. Der in diesem Fall gewählte Ausdruck der Ergebnisse anstelle der Anzeige ist nicht von der Logikart des Rechners, sondern nur davon abhängig, ob PROTAR eine Druckeinrichtung, bzw. einen Anschluß für einen Drucker besitzt oder nicht.

Unsere Programme „sitzen" also. Man kann aber, um zu mehr Sicherheit zu kommen, selbst Fehler bei der Programmeingabe herbeiführen, indem bewußt falsche Tasten gedrückt werden. Anschließend wird nach Rückstellen in den Rechen-Modus oder Run-Modus anhand einer Zahleneingabe beobachtet, wie der Rechner darauf reagiert.

4.4.3. Programmspeicherung

Wir unterstellen, daß wir es mit einem Rechner zu tun haben, der es ermöglicht, das so erarbeitete und auf Herz und Nieren geprüfte Programm auf einem Magnetstreifen zu speichern. Wir folgen den entsprechenden Anweisungen der Bedienungsanleitung unseres PROTARs und haben damit unsere Problemlösung auf „ewige Zeiten" konserviert. Stehen wir nun einmal erneut vor dem gleichen Problem, erinnern wir uns dieses kleinen Programmes, holen es wieder aus der Schublade, „laden" das Programm oder „lesen es ein", wann immer wir es benötigen und die gleiche Arbeit kann ohne weitere Vorbereitung wieder in der gleichen Weise ablaufen.

Haben Sie im vorliegenden Fall einen PROTAR, der nicht mit einer Schreib/Lese-einrichtung für Magnetstreifen ausgerüstet ist, dann müssen Sie die Befehlsfolge aus Ihren Unterlagen zum Programm erneut im Programm-Modus in den Rechner eintasten und somit benutzen; in diesem Fall ist der gleiche kurze Test durchzuführen. Denn es kann sich bei der Eingabe des Programms ein Fehler eingeschlichen haben. Davor ist niemand gefeit. Deswegen muß man nach jeder Programmeingabe die Richtigkeit derselben vor dem Rechnen testen. Außerdem sind vor Beginn der Datenverarbeitung die Datenspeicher zu löschen. Um das nicht zu vergessen, ergänzt man ein Programm meist mit einer sogenannten START-Routine. Das ist ein über einen eigenen Start-Label auszulösender Programmzweig, der beispielsweise das Löschen der Speicher für Sie übernimmt.

Wer überwiegend mit langen Programmen zu arbeiten hat und diese Programme noch dazu verhältnismäßig häufig benötigt, sollte die Mehrkosten für einen Rechner mit Magnetstreifen nicht scheuen. Neben einer Zeiteinsparung durch Vermeidung der neuen Eingabe des Programmes vor jeder Benutzung spricht auch die größere Sicherheit für die Speicherung der Programme auf Magnetstreifen. Ähnlich vorteilhaft sind Magnetstreifen-PROTARe für diejenigen Benutzer, die sehr viele unterschiedliche Aufgaben zu lösen haben, also sehr häufig ihre Programme wechseln müssen.

Schließlich ist es eine Frage des persönlichen Zeitmangels, ob er einen Rechner mit oder ohne Magnetstreifeneinrichtung benutzen will. Je kostbarer die Arbeitszeit des PROTAR-Benutzers, umso größer ist die Verschwendung, wenn er seine Programme immer wieder neu eingeben und testen muß, bevor er mit dem Rechnen beginnen kann.

4.4.4. Fehlerdiagnose

Test und Fehlerdiagnose gehören zusammen wie Pech und Schwefel und sind nicht voneinander zu trennen. Sie müssen zwangsläufig Hand in Hand ablaufen. Bei der Lösung komplexerer Probleme kann sich allerdings die Fehlersuche ausweiten. Nicht immer ist die Sache so einfach, wie in unserem in Abschnitt 4.4.2. dargestellten Fehlerbeispiel. Nicht immer bestätigen sich die Vermutungen über die Fehlerursache binnen so kurzer Zeit. Denn Fehler werden leider oft an ganz anderer Stelle im Programmablauf sichtbar, als ihre Wurzeln liegen. Dieser Umstand macht sich besonders bei längeren Programmen unangenehm bemerkbar. Ursache und Wirkung hängen nur sachlich, leider aber nicht örtlich zusammen.

Wie soll nun also bei der Fehlersuche vorgegangen werden? Wenn wir uns an die nachfolgende Reihenfolge halten, werden wir für die Behebung der Mehrzahl der Programmfehler gerüstet sein.

Beginnen wir bei jenen Programmstellen, bei denen Irrtümer erfahrungsgemäß am häufigsten auftreten, also bei den Verzweigungen. Vielfach finden wir die Fehlerursache in einer unzulänglichen oder völlig falschen Formulierung einer Bedingung. Der Wahrheitsgehalt der Abfrage ist neu zu überdenken und mit jenem zu vergleichen, der sich aus der verwendeten Abfrage ergibt. Die Behebung des Fehlers kann in der Umstellung der Abfrage auf das Gegenteil – „kleiner oder gleich" anstelle „größer" – oder in der Wahl einer anderen Abfrage bestehen.

Um die Richtigkeit bereits von Zwischenergebnissen zu prüfen, lassen wir uns diese anzeigen oder ausdrucken. Die dazu nötigen zusätzlichen Befehle für Programmstops, bzw. Ausdrucke fügen wir in unser Programm ein, vergessen aber nicht, diese Anweisungen nach erfolgreich verlaufenen Test wieder aus dem Programm zu eliminieren.

Eine weitere Möglichkeit der Kontrolle ergibt sich bei PROTARen mit Druckeinrichtung und der Möglichkeit der Auslistung der Programme auf einen Kontrollstreifen. Diese Rechner besitzen vielfach einen Schalter, der in eine besondere Stellung – TRACE genannt – gebracht werden kann. Bei Abarbeitung eines Programmes in Normalstellung des Schalters wird nur dann etwas am Kontrollstreifen angedruckt, wenn dies vom Programm her vorgesehen ist. In der Schalterstellung TRACE wird dagegen jeder einzelne Programmschritt, der durchlaufen wird, mit Zeilennummer und Kurzbezeichnung der Anweisung ausgedruckt. Verändert sich dabei die Anzeige, wird auch diese nach einer Veränderung durch eine automatisch durchgeführte Rechenoperation ausgedruckt. So können Sie den Rechenablauf schrittweise verfolgen und durch Abschätzen der Größenordnung des jeweiligen Zwischenergebnisses den Fehler diagnostizieren. Das gleiche können Sie im Prinzip auch bei Rechnern ohne Druckeinrichtung im Run-Modus durchführen. Sie arbeiten das Programm schrittweise mit Hilfe der SST-Taste ab. An der Veränderung der Anzeige können Sie wiederum auf die Fehlerursache schließen. Jeder Druck auf die SST-Taste bewirkt, daß das Programm um einen Schritt weitergeführt wird. Nur ist im allgemeinen nicht feststellbar, wo Sie sich dabei im Programm gerade befinden.

Lassen sich damit nicht alle Mängel erkennen, gehen Sie Schritt für Schritt das Programm von vorne durch und vergleichen die eingetasteten Befehle mit den aufgezeichneten. Dadurch läßt sich oft erkennen, daß auf das Eintasten einer Anweisung vergessen wurde. Hilft das nicht, so arbeiten Sie Ihre aufgezeichneten Anweisungen noch einmal durch und vollziehen im Gedanken den Programmablauf nach.

Erst wenn alle angeführten Hinweise zur Diagnose von Fehlern zu keinem befriedigenden Ergebnis geführt haben oder wenn sich zuviele Irrtümer eingeschlichen haben sollten, geht man daran, das Programm neu zu fassen und einzugeben. In manchen Fällen ist eine neue Durcharbeitung nach einer Zeit der Unterbrechung die wirtschaftlichere Lösung, als eine Sanierung des fehlerhaften Programmes.

4.4.5. Redigierung

Schon die Fehlerdiagnose war ein Teil der Redigierung: Wir haben unser Programm nach Fehlern abgesucht; eine vollständige Redigierung geht aber darüber hinaus.

Nach einer Weile der Benutzung eines Programms wird man feststellen, daß die eine oder andere Tatsache seinerzeit bei der Erstellung eines Programms nicht berücksichtigt wurde. Da haben wir gleich einen Bezug auf unser Übungsbeispiel, wo wir bereits festgestellt haben, daß wir eine Startroutine benötigen würden, mit der nach Beendigung einer Verarbeitung sofort mit einer neuen begonnen werden kann. Dazu müssen wir die beiden benutzten Summenspeicher 5 und 6 wieder auf Null stellen, also löschen. Daher legen wir beispielsweise einen neuen Label B fest und plazieren darunter folgende Tastenschritte in gleicher Weise für AL und UPN gültig:

LBL B

0

STO 5

STO 6

R/S

Vielleicht wechseln wir dabei auch die Labeln selbst und stellen die Startroutine, wie oben dargestellt, unter den Label A und die Verarbeitung selbst unter dem Label B. Das wäre zumindest formal besser. Man beginnt beim Label A und setzt mit Beginn der Datenverarbeitung beim Label B fort. An sich ist aber die Wahl der Label an keinerlei Beschränkungen gebunden. Die Redigierung beginnt also meist mit neuen Wünschen an das Programm und dabei meist mit einer Untersuchung, ob nicht mit Tricks Programmspeicherstellen eingespart werden können, da für die neuen Wünsche Platz geschaffen werden muß; es beginnt also der Kampf um die Programmspeicherplätze.

Daher kann es sein, daß ganze Programmzweige neu durchdacht werden müssen, um durch Umstrukturierung Programmspeicherplätze frei zu machen. Denn bei PROTAR ist die Zahl der zur Verfügung stehenden Programmspeicherplätze im allgemeinen fix und kann nicht nachträglich vergrößert werden. Bei größeren DV-Anlagen kann die Kapazität meist durch zusätzlichen Einbau von Speichern ausgeweitet werden, wir können das leider nicht. Dazu ist PROTAR auch nicht gebaut. Allerdings gilt hier eine kleine Einschränkung, da es heute zwei Rechner (s. S. 120/6) gibt, der aus dem gesamten Speicherangebot wenigstens eine Verteilung desselben in Programm- und Datenspeicher nach Gutdünken des Benutzers vornehmen läßt. Wir müssen daher hart arbeiten, um durch mögliche Umgruppierungen da und dort Speicher einsparen zu können. Die Lösung dieser Probleme liegt daher eindeutig in unserer Programmorganisation.

Wir gehen also Schritt für Schritt vor und stellen fest, daß

1. bestimmte Folgen von Anweisungen mehrmals vorkommen und prüfen dabei, ob sich diese mehrmals wiederholenden Befehlsfolgen in Form von Unterprogrammen so darstellen lassen, daß Programmspeicherplätze eingespart werden können;
2. ein Detailproblem zu umständlich gelöst wurde und die Lösung mit einer anderen Befehlsfolge mit weniger Programmschritten konstruiert werden kann;
3. gewisse Tastendrücke, z. B. die Fixierung von Dezimalstellen, überflüssig sind und daher eliminiert werden können.

Sicherlich waren wir im ersten Anlauf froh, einmal ein überhaupt funktionierendes Programm auf die Beine gestellt zu haben. Nun aber sind wir anspruchsvoller geworden und müssen daher ein Programm **überarbeiten**. Wir stellen auch unsere Forderungen an das Programm unter das Zeichen der Wirtschaftlichkeit. Alle diese Maßnahmen dienen dazu, ein **Programm zu optimieren**.

Ist nun einmal ein Programm nicht weiter zu vereinfachen und wollen wir trotzdem seine Funktion erweitern, bleibt uns nichts anderes übrig, als das Problem zu teilen und es auf zwei Programme aufzuteilen. Dies ist dabei oft die wirtschaftlichere Lösung, als einen größeren Rechner zu kaufen, da diese Fälle wohl selten auftreten.

Wir unterstellen also, daß wir nur selten an den Grenzen von PROTAR rühren wollen und legen es nur in den seltensten Fällen darauf an, ein Problem auf mehr als einen Magnetstreifen zu verteilen. Die Verarbeitung wird dadurch wohl etwas mühsamer, aber dies kann durchaus in Kauf genommen werden, wenn wir uns vor Augen halten, wie mühsam zuvor die Verarbeitung auf konventionelle Art und Weise war!

4.5. Programmdokumentation

Das Programm ist erstellt, der Test mit Hindernissen, aber letztlich doch erfolgreich verlaufen, Redigierung und Optimierung abgeschlossen. Die mit dem Programm zu rechnenden Aufgaben sind gelöst, das Programm ist für weitere Arbeiten bei gleicher Problemstellung einsatzbereit.

Um dessen Anwendung auch in der Zukunft zu erleichtern, müssen wir sämtliche mit der Programmerstellung zusammenhängenden Arbeiten dokumentieren. Eine fehlende Programmdokumentation wird sich besonders nach einer längeren Zeit der Nichtbenutzung unangenehm bemerkbar machen!

Damit das Programm immer richtig eingesetzt werden kann, benötigen wir eine **Bedienungsanleitung.** Spätere Änderungen und Ergänzungen werden durch das Vorhandensein einer **Programm-Listung,** die genaue Aufeinanderfolge sämtlicher Programmbefehle, wesentlich erleichtert. Wird unser Programm auch an anderer Stelle verwendet, orientiert sich der Benutzer anhand einer **Programmbeschreibung** über die Fähigkeiten des Programms. Vervollständigt wird die Dokumentation durch die Aufzeichnung einer Reihe von **Testbeispielen**. Diese werden vorrangig dann benötigt, wenn wir keinen Magnetstreifen-PROTAR verwenden und daher unser Programm vor jeder Verwendung neu eintasten müssen.

Die Dokumentation ist eine äußerst undankbare Aufgabe, auf die aber nicht verzichtet werden kann.

4.5.1. Bedienungsanleitung

Auch wenn es nur wenige Bemerkungen sind, die wir zur Bedienung des Programmes für erforderlich halten, wir zeichnen sie auf. Wir geben Schritt für Schritt an, die wir machen müssen, um unser Programm benutzen zu können. Es muß immer klar sein, welche Maßnahmen in welcher Reihenfolge getroffen werden müssen, die für die Benutzung des Programms von Bedeutung sind.

Mit „Schritt für Schritt" sind jetzt nicht die einzelnen Programmschritte gemeint, sondern die Vorgänge, mit denen eine Anwendung des Programmes kurz beschrieben werden.

Dazu nehmen wir wieder unser Übungsbeispiel, das der Preiserhöhungen. Dem Programm selbst wollen wir den Namen „Preis" geben. Bei der Formulierung der Bedienungsanleitung ist es auch ohne Belang, ob wir nun das in der AL- oder UPN-Logik geschriebene Programm meinen.

Bedienungsanleitung für das Programm PREIS

1. Programm PREIS einlesen.
2. Programm zur Löschung der Datenspeicher mit der Taste A starten.
3. Eingabe eines Preises mit der Taste B; der Rechner bleibt mit der Anzeige des erhöhten Preises stehen.
4. Letzter Preis?
   4.1. Wenn ja, dann Eingabe der Zahl 0 mit der Taste B und Abruf der Ergebnisse. In der genannten Reihenfolge werden ausgegeben:
   - Anzahl der um 5% erhöhten Preise;
   - Anzahl der um 6% erhöhten Preise;
   - Gesamtzahl der erhöhten Preise.

   4.2. Wenn nein, Fortsetzung bei Schritt 3.

Achtung! Im Falle eines Tests ist eine Verarbeitung bei Schritt 2 zu beginnen!

Wir unterstellen, daß wir das Programm bereits ergänzt und geändert haben.

An dieser Stelle sei kurz der Fachausdruck „Einlesen" erläutert: Das Einschieben des Magnetstreifens wird als „Einlesen" bezeichnet. PROTAR liest nach dem Einschieben des Magnetstreifens mit Hilfe des Schreib-/Lesekopfes gewissermaßen die Anweisungen und befördert sie in sein „Gedächtnis", den Programmspeicher. Das Einspeichern der Anweisungen wird auch häufig als „Laden des Programmes" bezeichnet.

Mit der gleichen Vorrichtung wird umgekehrt ein Magnetstreifen mit Informationen aus dem Rechner, wie Programmanweisungen und Daten, „beschrieben".

				„PREIS"
Start	Eingabe P 0 = Abruf			

Falls wir einen Magnetstreifen-PROTAR benutzen, müssen wir ferner den Programmstreifen beschriften, damit wir ihn bei Bedarf ohne Schwierigkeiten auffinden können. „Leere" Magnetstreifen werden von den Herstellern der Rechner in albenähnlichen Kartentaschen geliefert. Diese Kunststofftaschen dienen dann auch der Aufbewahrung der Magnetstreifen für Programme und Daten. An einer passenden Stelle tragen wir den Namen des Programmes ein. Die beiden Label-Stellen A und B versehen wir mit entsprechenden Hinweisen, aus denen auf die mit diesen Tasten auszulösenden Funktionen ohne Schwierigkeiten geschlossen werden kann.

In der obigen Skizze werden die fünf Kästchen der unteren Reihe für die Kennzeichnung der Primär-Labeln A, B, C, D und E, die oberen fünf Kästchen sind der Kennzeichnung der Sekundär-Labeln A', B', C', D' und E', bzw. a, b, c, d und e vorbehalten.

Da wir in unserem Übungsbeispiel nur zwei Labeln benötigen, brauchen wir auch nur zwei Kästchen beschriften und setzen den Namen des Programmes oben rechts auf den Magnetstreifen. Vor den Programm-Namen können wir auch noch eine Programm-Nummer setzen. Dies empfiehlt sich bei einer größeren Anzahl von Individualprogrammen, womit auch eine übersichtliche Archivierung ermöglicht wird.

4.5.2. Programm-Listung

Nehmen wir an, daß mit einem Magnetstreifen irgend etwas passiert ist, so daß er vom Rechner nicht mehr angenommen wird. Sie können Ihr Programm nicht mehr einlesen. Wie wollten Sie weiterarbeiten, wenn Sie keine Programm-Listung hätten? Die wichtigste

Listung ist jene, die Sie auf Ihrem Codierblatt vor sich liegen haben. Zusätzlich sind die mit Druckern ausgerüsteten PROTARe heute in der Lage, auch über den Drucker eine lückenlose Aufstellung der aufgezeichneten Programmschritte zu liefern. Diesen Streifen heftet man dann zu den übrigen Programmunterlagen. Mit dieser Einrichtung kann man übrigens auch eine handschriftliche Auflistung der Anweisungen mit den im Rechner gespeicherten vergleichen. Ein wertvolles Hilfsmittel bei Fehlersuche und Redigierung!

In den von Hand erstellten Codierblättern, das sind jene, in die Rechenanweisungen eingetragen werden, sollte man an passenden Stellen Kommentare einfügen, um auch später zu wissen, warum man diesen oder jenen Befehl gewählt hat. Das Programm wird für spätere Redigierung leichter lesbar. Die Angabe des Grundes für verschiedene Maßnahmen kann, soweit dieser angegeben ist, spätere Korrekturen und Änderungen wesentlich erleichtern und beschleunigen.

Wenn Sie zurückblättern, haben wir ja bei jeder Darstellung unseres Übungsbeispiels in den einzelnen Tastenfolgen schon so gehandelt, ohne uns über das „Warum?" weitere Gedanken zu machen.

4.5.3. Programmbeschreibung

In der Programmbeschreibung oder auf gesonderten Beilagen sollte man sämtliche verwendeten Formeln und Berechnungsgrundlagen darstellen und erläutern. Die in den Formeln verwendeten Symbole sind zu erklären. Als Autor eines Programms möchte man später selber noch genau wissen, welche Bedeutung eine getroffene Maßnahme hat. Im übrigen muß man allzu oft ein Programm auch anderen erklären, die damit zu arbeiten haben.

Zur Programmbeschreibung gehört selbstverständlich auch das Ablaufdiagramm, das mitunter noch besser – weil graphisch dargestellt – den Programmablauf dokumentieren kann. Oftmals genügt auch die Darstellung eines komplizierteren Details, um völlige Klarheit zu schaffen.

4.5.4. Testbeispiele

Zu jeder Dokumentation gehört zumindest ein Testbeispiel. Mit diesem ist es auch möglich, Störungen im Rechner festzustellen, wenn ein Programm, obwohl richtig aufgezeichnet, nicht die erwarteten Ergebnisse bringt. Die von den Herstellern fallweise mitgelieferten Diagnose-Programme helfen hier in den meisten Fällen. Wer aber wird vor dem Einsatz neuer Programme als erstes die Funktionstüchtigkeit des Rechners überprüfen? Das Diagnose-Programm wird wohl erst in hartnäckigen Fällen und als letztes eingesetzt, bevor PROTAR als ganzes der Funktionsuntüchtigkeit bezichtigt werden kann. Dann ist der Rechner zur Reparatur der nächstgelegenen Service-Werkstätte zu senden.

Wenn ein Programm relativ selten benutzt wird, dient es der eigenen Beruhigung, vor einer Verarbeitung den Test zu wiederholen. Dabei gewinnt man auch wieder etwas Übung in der Handhabung des Programms. So kann man sich auch vor etwaigen Fehlbedienungen schützen. Erst wenn der Test erfolgreich verlaufen ist, sollte man zur Datenverarbeitung schreiten. Sind aber Fehler aufgetreten, ist eine Fehlerdiagnose durchzuführen und die sich daraus ergebenden Korrekturen sind vorzunehmen. Es kann übrigens durchaus vorkommen, daß man schon früher ein Programm nicht in der korrigierten Form aufgezeichnet hat, sondern nur in der ursprünglichen.

Im Test sollten sämtliche denkbaren Varianten, die das Programm bewältigen kann, mit Beispielen belegt werden. Denn nichts ist während des Rechnens unangenehmer, als mit einer neuen, nicht dokumentierten Variante konfrontiert zu werden und damit die vorliegende Aufgabe nicht sofort lösen zu können. Im besonderen sollte der Behandlung aller Grenzfälle breiter Raum gewidmet werden.

4.6. Änderungsdienst

Dieser Aufgabenbereich ist ebenso undankbar, wie die Programmdokumentation. Aber auch er gehört zu den wichtigsten Dingen einer Programm-Organisation. Und weil auch er nicht zu den kreativen Tätigkeiten zu zählen ist, wird er ebenso gerne und leicht vernachlässigt. Dieses Thema hat weder mit der DV noch dem PROTAR etwas zu tun.

Schon mehrfach wurde angedeutet, daß im Verlaufe der Anwendungen von Programmen mit ziemlicher Sicherheit mit Änderungen zu rechnen ist; zum Beispiel bestimmte Rechenabläufe betreffend, die bisher wohl fehlerfrei, aber doch nicht ganz zur vollen Zufriedenheit des Benutzers abgelaufen sind, weil rein organisatorisch gesehen in der Praxis manches anders angetroffen wurde. Aber das ist halb so schlimm, wenn wir aus dieser unvermeidlichen Tatsache auch die Konsequenzen ziehen: Änderungen vornehmen – aber auch dokumentieren!

Nehmen wir für unser Übungsbeispiel an, wir hätten vergessen, die letzte Änderung, den Einbau einer Startroutine und den Wechsel der Label A und B zu dokumentieren. Es würde einige Zeit dauern, bis wir uns zurecht finden und wertvolle Arbeitszeit ginge verloren, wahrscheinlich mehr, als wir durch Vernachlässigung des Änderungsdienstes bewußt oder unbewußt als Zeitvorteil hatten herausholen können.

Die Änderungen sind in allen davon betroffenen Unterlagen durchzuführen, und zwar:

- auf der Magnetkarte,
- in der Programm-Listung,
- im Ablaufplan,
- in der Bedienungsanleitung und gegebenenfalls
- in der Programmbeschreibung.

Dabei sollte nicht vergessen werden, die Unterlagen bei allen Mitarbeitern, die das Programm benutzen, zu korrigieren.

Das geänderte Programm erhält nach der Programm-Nummer den Buchstaben A, einen Schrägstrich und die Zahl 1 oder einen anderen Hinweis, daß das vorliegende Programm sich aus einem ähnlichen herleitet und somit nur die letztgültige Modifikation darstellt. Für diesen Fall ist es ratsam, das Programm auch in seiner ursprünglichen Form aufzubewahren, da nicht ausgeschlossen scheint, daß man auch einmal den umgekehrten Weg geht und zu einer älteren Programm-Version zurückkehrt.

Vergessen Sie bei einer Änderung nicht das Datum zu vermerken, damit man beim Vergleich von Ergebnissen aus unterschiedlichen Programm-Versionen immer weiß, woher man die Ergebnisse genommen hat. Auch auf die ursprüngliche Dokumentation gehört natürlich schon ein Datum.

Wir haben versucht, ein Fachgebiet, zumindest aus der Sicht der PROTARe, einem neuen Interessentenkreis zu erschließen. Dabei konnte es natürlich weniger darauf ankommen, Unterschiede einzelner Rechner zu erläutern, als die Organisation der Programmierung in Augenschein zu nehmen. Die Geräte selbst werden in den jeweiligen Bedienungsanleitungen hinreichend beschrieben. Unsere Aufgabe war es, diese Anleitungen durch die Behandlung der vom jeweiligen PROTAR-Modell unabhängigen und daher allgemein gültigen Grundsätze der Programmierung zu ergänzen.

4.7. Hinweise für Fortgeschrittene

Mit der Fertigkeit im Umgang mit PROTARen wachsen auch die Anforderungen, die der Routinier an sein Gerät stellt. Deswegen werden im folgenden eine Reihe von Sonderfällen der Programmierung behandelt.

Nicht jeder dieser Sonderfälle kann auf allen PROTARen nachvollzogen werden. Deswegen wird im einzelnen auf die Anwendungsmöglichkeiten verwiesen.

Weitere Hilfen in der Lösung von Sonderfällen bieten teilweise auch die Bedienungsanleitungen. Besonders zu empfehlen ist aber die Lektüre periodisch oder unregelmäßig erscheinender Kundendienstzeitschriften der Hersteller. Unter Zuhilfenahme dieser Zusatzinformationen in Verbindung mit den Erfahrungen aus der eigenen Praxis wird sich ein persönlicher, mit verschiedenen „Programm-Spezialitäten" angereicherter Programmierstil entwickeln.

Der Neuling wird diesen Abschnitt vorläufig übergehen und auf ihn erst dann zurückgreifen, wenn er bereits einige Praxis im Umgang mit PROTAR gewonnen hat.

4.7.1. Mehrfachbelegung von Speichern

Diese „Feinheit" wollen wir bewußt an den Beginn unserer Sonderfälle stellen. Einerseits, weil diese Möglichkeit auf allen PROTARen genutzt werden kann, sofern sie die Tasten INT und FRAC, bzw. INV INT aufzuweisen haben und andererseits, weil man speziell im kommerziell-betriebswirtschaftlichen Bereich „nie genug Speicher haben kann".

Es wurde bereits mehrfach auf diese Möglichkeit hingewiesen, hier wollen wir darstellen, wie sie in der Praxis verwirklicht werden kann.

Selbstverständlich sollte die **gleichzeitige Mehrfachverwendung von Datenspeichern** nur bei echtem Mangel an Speicherplätzen angewendet werden. Denn das Einordnen und Herauslösen einzelner Teildaten bedingen bestimmte Programmfolgen. Dadurch erhöht sich der Bedarf an Programmspeicherplätzen. Das Abspeichern und Abrufen ist nun nicht mehr allein durch das Drücken der STO-, bzw. RCL-Taste möglich, sondern bedarf jeweils eines bestimmten Programmzweiges. Wir untersuchen vorerst einmal den Charakter der zu speichernden Werte.

Sind es sogenannte Fixdaten, d. h. Daten, die sich im Verlaufe der Rechenoperationen nicht oder zumindest nicht in der benötigten Stellenzahl ändern, haben wir die ersten geeigneten Objekte für unser Vorhaben. Wollen wir aber unter Umständen sich ändernde Werte auf diese Art und Weise abspeichern, müssen wir vorher genau prüfen, wie groß, bzw. wieviele Stellen maximal ein Ergebnis haben wird.

Es muß unter allen Umständen vermieden werden, daß die Werte letztlich doch größer werden, als man es ursprünglich geplant hat. Die Folge davon ist, daß durch höhere Stellenzahlen entweder Stellen verloren gehen, oder die übergreifenden Stellen in eine andere Zahl hineingreifen und diese unwiederbringlich verändern und damit unbrauchbar machen. Daher ist bei Abspeicherung sogenannter **Bewegungsdaten in der geplanten Form äußerste Vorsicht** geboten!

Diese Praxis der gleichzeitigen Belegung eines Datenspeichers mit mehreren Zahlen sei an einem praktischen Anwendungsbeispiel dargestellt.

Bei der Berechnung der Wiederbeschaffungswerte im Zusammenhang mit der Ermittlung der kalkulatorischen Kosten benötigen wir unterschiedliche Aufwertungsfaktoren für unterschiedliche Anlagegüter. Dazu wollen wir auch den Prozentsatz der Kapitalverzinsung unterbringen. An Anlagegruppen hätten wir das Gelände, die Bauten nach einem Baukostenindex und maschinelle Anlagen nach einem Maschinenkostenindex, die unterschiedlich aufzuwerten wären. Wir haben es hier mit vier Prozentsätzen zu tun, die während der gesamten Verarbeitung unverändert bleiben. Sie eignen sich daher ausgezeichnet für unsere **gepackte Darstellung** in einem Datenspeicher. In einem Speicher kann man im allgemeinen 8 bis 12 Stellen abrufbar unterbringen. Für unsere Prozentsätze

wählen wir maximal zweistellige und ganzzahlige Werte, sie erhalten auch eine Bezeichnung:

p_G ..... Aufwertung Gelände mit 12%,
p_B ..... Aufwertung Bauten mit 10%,
p_M ..... Aufwertung maschinelle Anlagen und Formen mit 6%,
p_K ..... Kapitalverzinsung mit 8%.

Im Speicher 0 sollen diese Prozentsätze gemeinsam abgespeichert werden; dazu wählen wir folgende Darstellung:

.12100608

Die vier Prozentsätze setzen wir nach dem Dezimalpunkt einfach hintereinander. Dabei berücksichtigen wir die den Zahlen 6 und 8 vorangehenden Nullen.

Wie gelangen diese Werte in den Speicher? Am einfachsten, wir tasten die vier Prozentsätze in der oben dargestellten Form ein und speichern sie ab. Dies erfolgt bekanntlich mit der Tastenfolge STO 0. Man kann aber auch den Weg über das Programm wählen. Dies wird vor allem dann von Vorteil sein, wenn wir nur einen der vier Werte ändern wollen. In der AL-Logik erfordert dieser Vorgang einen Hilfsspeicher. In der UPN-Logik übernehmen die Stack-Speicher diese Funktion. Die einzelnen Teile des Inhaltes des Speichers 0 werden in der Anzeige aufbereitet und im Hilfsspeicher zusammengesetzt. Anschließend daran wird die geänderte Zahlenkette vom Programm her im Speicher 0 abgespeichert.

Wir wollen den an zweiter Stelle stehenden Baukostenindex von 10 auf 12% abändern und schreiben dieses Programmbeispiel in der AL-Logik. Als Eingabelabel wählen wir die Taste B:

LBL B	Eingabelabel für den Baukostenindex
÷	Einleitung der Division durch 10.000; damit wird der neue Wert an die gleiche Stelle in der Zahlenkette gebracht, an der der alte derzeit noch steht.
1	
EE	
4	1 EE 4 bedeutet 10^4, somit 10.000; angezeigt wird diese Zahl in wissenschaftlicher Darstellung und zwar 1 04. Diese Zahlendarstellung wurde im Abschnitt 2.3.2. erläutert. Mit dem Drücken der EE-Taste wird gleichzeitig die wissenschaftliche Zahlendarstellung angewählt.
=	Anzeige 1.2-03
STO 9	Abspeicherung der Zahl 0.0012 in den Hilfsspeicher 9
RCL 0	Abruf der Zahlenkette .12100608 zur Isolierung des nicht zu verändernden Wertes p_G
x	Einleitung der Multiplikation mit 100
1	
EE	
2	1 EE 2 bedeutet 10^2, somit 100
=	Anzeige 1.2100608 01; das ist 12.100608
INT	Bildung des ganzzahligen Teiles der angezeigten Zahl; Anzeige 1.2 01; das ist 12

÷	Einleitung der Division durch 100
1	
EE	
2	
=	Anzeige 1.2-01; das ist 0.12
SUM 9	Aufsummierung zur neuen Zahlenkette im Hilfsspeicher 9; sein Inhalt ist im Augenblick 0.1212
RCL 0	Abruf der alten Zahlenkette zur Isolierung der ebenfalls nicht zu verändernden Werte p_M und p_K
x	Einleitung der Multiplikation mit 10.000
1	
EE	
4	
=	Anzeige 1.2100608 03; das ist 1210.0608
INV INT	Bildung des Dezimalteiles der angezeigten Zahl mit 0.0608; in der Anzeige würde allerdings 6.08-02 stehen
÷	Einleitung der Division durch 10.000
1	
EE	
4	
=	Anzeige 6.08-06; das ist 0.00000608
SUM 9	Aufsummierung zur neuen Zahlenkette in den Hilfsspeicher 9
RCL 9	Abruf der neuen Zahlenkette und . . .
STO 0	. . . Abspeicherung in den ursprünglichen Speicher; die alte Zahlenkette wird dabei überschrieben und ist somit verloren.
R/S	Stop und Anzeige 1.2120608-01; das ist 0.12120608.

Wir können die inzwischen eingetretene nicht an eine bestimmte Rechenlogik gebundene wissenschaftliche durch die Betätigung der EE-Taste ausgelöste Zahlendarstellung wieder rückgängig machen, indem wir vor dem Stopbefehl die Tastenfolge INV EE einfügen, d. h. die Umkehrung der wissenschaftlichen in die normale Darstellung vorsehen. Im Gegensatz zu Rechnern mit der UPN-Logik ist hier die EE-Taste nicht nur die Exponententaste für die Basis 10, sondern auch gleichzeitig die Taste, mit der die wissenschaftliche Darstellung von Zahlen angewählt werden kann. Warum die an sich auch mögliche Durchführung von Multiplikationen oder Divisionen unter Verwendung der **10^x-Taste nicht gewählt werden darf**, wurde bereits im Abschnitt 2.3.2. eingehend erörtert. Aber auch das Gegenteil sei noch besprochen, nämlich der Abruf eines bestimmten Prozentsatzes. Auch dieser wird in der AL-Logik dargestellt; dem bereits geübten Leser wird es wenig Schwierigkeiten bereiten, sich das gleiche für die UPN-Logik, für die ein etwas anderer Ablauf gilt, zu erarbeiten.

Als Abrufbeispiel wollen wir die Isolierung des Aufwertungssatzes für maschinelle Anlagen wählen. Dieser Wert steht an dritter Stelle in der Zahlenkette der vier zweistelligen Aufwertungsprozentsätze. Da ein Abruf in der Regel anläßlich einer Berechnung vom Programm aus erfolgt, wählen wir hier den Ziffernlabel 3, mit dem wir den Beginn des Unterprogrammes für diesen Abruf markieren:

LBL 3	Abruf und Isolierung p_M
RCL 0	Abruf der Zahlenkette 0.12120608; wir unterstellen, daß es sich dabei um die im vorigen Übungsbeispiel geänderte handelt.
x	Einleitung der Multiplikation mit 10.000
1	
EE	
4	
=	Anzeige 1.2120608 03; das ist 1212.0608. Auch hier verbleibt die Anzeige, durch die Betätigung der EE-Taste bedingt, in der wissenschaftlichen Darstellung.
INV INT	Bildung des Dezimalteiles der angezeigten Zahl mit 0.0608; in der Anzeige würde allerdings 6.08-02 stehen.
x	Einleitung der Multiplikation mit 100
1	
EE	
2	
=	Anzeige 6.08 00; das ist ebenfalls 6.08, jedoch ohne die den Exponentialteil markierenden beiden Nullen.
INT	Bildung des ganzzahligen Teiles der angezeigten Zahl
INV EE	Rückführung der Zahlendarstellung in die normale Form
INV SBR	Stop und Anzeige 6.

Wie wir dann in der Praxis im Programm weiterrechnen, mit der Zahl 6 oder ihrem 100. Teil, richtet sich nach der gewählten Rechenformel. Es ist mir hier im wesentlichen nur auf die Darstellung der **Dezimalpunktverschiebung** im Zusammenhang mit der Verwendung der Tasten zur Berechnung des ganzzahligen Teiles, bzw. des Dezimalteiles eines Wertes angekommen und auf die Möglichkeit der Abspeicherung und des Abrufs aus einer Zahlenkette. Die übrigen Varianten der Änderung und des Abrufs eines Prozentsatzes wird sich der Leser leicht selbst erarbeiten können.

4.7.2. Datenorganisation

Es gibt auch noch einen anderen Weg, Datenspeicher zu sparen. Vorausgeschickt sei, daß die nun zu besprechende Methode nur bei jenen Rechnern realisiert werden kann, bei denen es möglich ist, Teile von bestehenden Programmen durch neue Programmteile zu ersetzen, ohne daß der verbleibende Programmteil beeinflußt wird. Damit ist schon angedeutet, daß es sich in dem Falle um „Datenspeicher" handelt, die in Programme eingebettet sind. Die Daten werden also durch ein Programm erzeugt.

Ein bestehendes Programm wird ab einem bestimmten Punkt durch ein neues Programm überschrieben, wobei der vor diesem Punkt liegende Teil im Programmspeicher erhalten bleiben soll. Dieses auf solche Weise angehängte Programm enthält nun jene Daten, die wir noch zusätzlich brauchen und die nicht mehr in den Datenspeichern untergebracht

werden können. Nachdem die Daten also in einem Programm eingebaut sind, könnten wir in diesem Fall auch von einem **Datenprogramm** sprechen.

Wie nutzt man nun diese von verschiedenen PROTARen gebotene Einrichtung?

Das ursprüngliche Programm, das beim Neueinlesen eines Programmes erhalten bleibt, wird als **Steuerprogramm** oder **Operationsprogramm** verwendet, und die nacheinander einzulesenden Programme werden als **Datenprogramme** aufgebaut. Von dieser Konstruktion wurde z. B. bei der Erstellung einer automatischen Kalkulation Gebrauch gemacht.

Im **Operationsprogramm**, das einen Teil des Programmspeichers belegt, wird der Ablauf einer Kalkulation gesteuert. Dieses Programm ist gewissermaßen das Abbild unseres Kalkulationsschemas.

In den **Datenprogrammen** haben wir die Stammdaten unserer Standardartikeln festgehalten. Diese Stammdaten sind durch Sprunganweisungen in Unterprogramme voneinander getrennt, wobei diese Unterprogramme wiederum Teile des Operationsprogramms sind.

Unsere Verrechnungsdaten befinden sich nun in den Datenspeichern und werden durch die im Operationsprogramm befindlichen Unterprogramme der Reihe nach aufgerufen.

Eines ist klar: Damit lassen sich sicherlich nur wenig rechenintensive Programme bearbeiten; eine Kalkulation ist dabei das beste Beispiel hierfür und steht nur als eine Möglichkeit für mehrere im kommerziell-betriebswirtschaftlichen Bereich. In diesen ist der Rechenaufwand im allgemeinen nicht sehr groß. Die Datenprogramme sind – sehen wir einmal von den als Anweisungen gespeicherten Daten ab – völlig identisch. Die Daten werden nun an den benötigten Stellen einfach zwischen die Sprunganweisungen nach Unterprogrammen eingefügt. Dies bewältigt am besten mit einem sogenannten **Datenprogramm-Generator**. So wird es auch auf relativ einfache Weise möglich, Magnetstreifen zu einer Art Magnetkonten umzufunktionieren.

Für diese Technik läßt sich kein Zahlen- oder Programmdetail bieten, weil dies den Rahmen der vorliegenden Erläuterungen vom Umfang her sprengen würde. Im Anhang wird in den Beispielen A und B im übrigen eine normale und im Anschluß daran auch automatische Kalkulation noch eingehender besprochen.

Mit der Datenorganisation sollte nur besonders darauf hingewiesen werden, Daß es Mittel und Wege gibt, unter bestimmten Voraussetzungen die Möglichkeiten der Datenspeicherung bedeutend zu erweitern.

Bei den Beispielen A und B haben wir in 26 Datenspeichern mit 260 Speicherstellen 80 Verrechnungsdaten unterbringen können. Für den Datenprogrammen kann gerechnet werden, daß auf diesem Wege noch etwa 20 bis 25 Artikelstammdaten mit ungefähr 80 bis 100 Ziffern unterzubringen sind. Die genaue Zahl hängt von der Anzahl der Programmspeicherstellen ab, die für das Grundprogramm, das wir als Datenprogramm-Generator bezeichnet haben, benötigt werden.

4.7.3. Mehrfachverwendung einer freien Funktionstaste

Bei allen Rechnermodellen ist es möglich, einer freien Funktionstaste, das sind die bereits mehrfach erwähnten Alpha-Tasten, **mehr als eine Funktion** zuzuweisen. Diese Mehrfachbelegung ist allerdings mit Abfragen gekoppelt, da verständlicherweise eine Mehrfachverwendung auf das Vorhandensein mehrerer durch eine Taste erreichbarer Programmzweige schließen läßt.

Theoretisch könnten mit den Ziffern 0 bis 9 einer Taste 10 Möglichkeiten unterlegt werden. Es ist klar, daß damit die Verwendung dieser Ziffern als Eingaben ausgeschlossen ist. Meist beschränken wir uns dabei auf den Wert 0.

Ohne davon besonders Kenntnis zu nehmen, haben wir uns dieser Möglichkeit bereits in Verfolgung des Übungsbeispieles in dem Programm bedient, das uns unterschiedliche Preiserhöhungen rechnen sollte, wobei zusätzlich die Anzahlen der umgerechneten Preise zu ermitteln waren. In diesem im Abschnitt 3.4.3. besprochenen Beispiel haben wir mit der Eingabe der Zahl 0 den Abruf der Anzahlen der umgerechneten Preise angesteuert. Die erste Abfrage in dem Programm galt der Kontrolle der Eingabe auf den Wert 0. Wie dabei programmtechnisch vorzugehen war, wurde im Abschnitt 4.3. erläutert.

Wir können uns dieser Möglichkeit aber auch in einem anderen Fall bedienen. Mit einer einzigen Taste soll sowohl eine Zahleneingabe, als auch der Abruf eines gespeicherten Wertes erfolgen. Man kann auch diesen Vorgang mit der Zahl 0 steuern, vorausgesetzt, diese Zahl wird nicht als Eingabe für eine Berechnung verwendet. Für diese Mehrfachbelegung einer freien Funktionstaste gibt es Rechner (s. S. 120/3), die dafür besonders geeignet sind. Auf diesen Rechnern ist auch die folgende Technik zu verwirklichen.

4.7.4. Eingabe während einer Pause

Der Sinn dieser Technik liegt darin, während der Zeit, in der der Rechner zur Anzeige eines Wertes etwa eine Sekunde lang anhält, diese zur Eingabe einer Zahl zu benutzen und sich das Drücken einer Label- oder der R/S-Taste für einen neuerlichen Start zu sparen. In diesem Fall läuft das Programm in einer „ewigen Schleife", aus der es nur herauskommt, wenn eine Zahl eingegeben wird. Damit wird man auch vom Zeitpunkt der Eingabe unabhängig, da das Programm beim Durchlauf der ewigen Schleife sozusagen in „Wartestellung" geht und es dem Benutzer anheimstellt, den Zeitpunkt der Eingabe ganz nach freiem Gutdünken selbst zu wählen.

Diese Einrichtung besteht in dem Faktum, daß anläßlich einer Zahleneingabe in den Rechner über das Tastenfeld, bzw. des Einlesens von Daten mit Hilfe eines Magnetstreifens ein bestimmter Flag gesetzt wird. Diese Flag hat noch eine weitere bemerkenswerte Eigenschaft: Bei der Abfrage, ob dieser Flag gesetzt ist, wird er automatisch gelöscht. Diese Tatsache macht man sich zunutze, die Eingabe einer Zahl während einer etwa eine Sekunde dauernden Pause vornehmen zu können. Wie dies programmtechnisch abläuft wird im folgenden erläutert.

	Das Programm schreitet voran und kommt zum Label 2.
LBL 2	Zu diesem Label wird zurückgesprungen, im Fall daß während der Programmpause keine Zahl eingegeben wurde.
CF3	Der Flag 3 wird sicherheitshalber gelöscht, da unter Umständen mit seiner Existenz im Verlaufe des Programms zu rechnen war.
PAUSE	Das Programm hält hier kurz an. Wird eine Zahl eingegeben, verlängert sich die Programmpause nach jedem einzelnen Druck auf eine Zifferntaste, sodaß auch längere Zahlen ohne Schwierigkeiten eingegeben werden können. Bei dieser Eingabe wird der Flag 3 gesetzt.
F3 ?	Nun folgt die Abfrage, ob dieser Label 3 tatsächlich gesetzt wurde.
GTO 5	Ist die Frage mit ja zu beantworten, wird zum Label 5 gesprungen und dort die Berechnung fortgesetzt.
GTO 2	War die Frage zu verneinen, springt das Programm zum Label 2 zurück, um erneut auf die Eingabe einer Zahl zu warten. Hier befindet sich das Programm in einer ewigen Schleife, siehe dazu auch Abschnitt 3.4.3.
LBL 5	Nach einer Eingabe wird an dieser Stelle im Programm fortgesetzt

Es gibt auch eine zweite Möglichkeit, während einer Programmpause eine Zahl einzugeben. Im zuvor dargestellten Programmdetail wird vor dem Label 2 die Zahl 0 gesetzt. Die Abfrage wird nun nicht nach dem Flag 3 gestellt, sondern darnach, ob in der Anzeige der Wert 0 steht.

Wird diese Frage mit ja beantwortet, springt das Programm zum Label 2 zurück und befindet sich damit ebenfalls in einer ewigen Schleife, aus der es erst bei einer von 0 verschiedenen Eingabe herausfindet. Eine Zahl 0 dürfen wir nicht als Zahleneingabe für weitere Berechnungen vorsehen. Im Prinzip können wir für die Steuerung der Eingabe einer Zahl während einer Programmpause jeden Wert verwenden, wenn wir diesen mit Sicherheit nicht für eine Eingabe benötigen. Die Tastenfolge für das erörterte Programmdetail sieht folgendermaßen aus:

0	In die Anzeige setzt der Rechner den Wert 0.
LBL 2	Dieser Label markiert wiederum den Beginn der ewigen Schleife, zu dem im Falle einer fehlenden Eingabe zurückgesprungen wird.
PAUSE	Das Programm hält hier kurz an. Wird während der Pause kein von 0 verschiedener Wert eingetastet, verbleibt 0 in der Anzeige. Erfolgt aber eine entsprechende Eingabe, wird der Wert 0 in den Y-Speicher des Stacks transferiert und in der Anzeige, also im X-Speicher, befindet sich die eingegebene Zahl.
x = 0	Der X-Speicher wird auf den Wert 0 abgefragt.
GTO 2	Bei der Antwort ja wird zum Label 2 zurückgesprungen. Bei der Antwort nein wird der Rücksprungbefehl ignoriert und an dieser Stelle im Programm fortgesetzt.

Auch diese Einrichtung funktioniert leider nicht bei allen PROTARen, die mit einer PAUSE-Taste ausgestattet sind. Diese Eingabesteuerung ist ein recht brauchbares Hilfsmittel in der Programmgestaltung.

Die PAUSE-Einrichtung kann auch zur Eingabe von Folgeprogrammen benutzt werden. Dies wird dann der Fall sein, wenn eine Problemlösung so umfangreich ist, daß das daraus resultierende Programm auf mehr als einem Magnetstreifen gespeichert werden mußte. Die Eingabe des Folgeprogrammes kann nun analog der zuvor beschriebenen Zahleneingabe ebenfalls während einer Programmpause erfolgen. Dieser Vorgang ist ebenfalls nur bei jenen Rechnern möglich, auf denen die Zahleneingabe während einer Programmpause realisierbar ist. Diese Möglichkeit wurde aber nur der Vollständigkeit halber erwähnt. In der Praxis ist aber zu empfehlen, diesen Programmwechsel während eines Programmstops vorzusehen und sich allenfalls dabei eine Zahl anzeigen zu lassen, die auf die Notwendigkeit der Eingabe des Folgeprogrammes hinweist.

4.7.5. Flag-Logik

Diese Art von Logik hat mit dem Rechensystem, in dem unser PROTAR arbeitet, nichts zu tun. Anhand eines Beispieles sei gezeigt, was damit gemeint ist. Für diese Übung wählen wir einen in der UPN-Logik arbeitenden PROTAR.

Hauptprogramm	**Unterprogramm zur Aufrechnung der Zahl 5**
●	●
S	●
●	●
F1?	LBL 0
GSB 0	5
●	+
●	RTN
●	●
	●
	●

Diese bruchstückartige Darstellung eines Programmteiles besagt folgendes: An einer bestimmten Stelle im Hauptprogramm wird abgefragt, ob der Flag 1 gesetzt ist. Nur in diesem Fall ist dem augenblicklich im X-Register, also in der Anzeige befindlichen Wert die Zahl 5 zuzuzählen und dann wieder normal im Programm fortzufahren. Ist der Flag 1 nicht gesetzt, wird die Anweisung GSB 0 übersprungen und ohne diese Aufrechnung weitergerechnet. Dieses Problem kann aber auch sparsamer bezüglich der Verwendung der Programmspeicher und Rechenzeit gelöst werden:

```
●
●
●
F1 ?
5
F1 ?
÷
●
●
●
```

Hier passiert das gleiche: Das Programm kommt zur ersten Abfrage und stellt fest, daß der Flag 1 gesetzt ist, setzt dann die Zahl 5 in den X-Speicher, während der letzte Wert dabei in den Y-Speicher zurückgereiht wird, kommt zur gleichen Abfrage noch einmal und da die Antwort wieder ja ist, wird die Addition ausgeführt und dann normal weitergerechnet. Ist die Antwort nein, wird bei beiden Abfragen der jeweils nächstfolgende Befehl, das Setzen der Zahl 5 und die Additionsanweisung übersprungen und, ohne die Zahl in der Anzeige zu verändern, weitergerechnet.

Die Flag-Logik ist auch in der Lage, Probleme zu bewältigen, die mit Kombinationen von Bedingungen entstehen. Folgende Kombinationen sind denkbar:

Sprünge im Programm werden ausgeführt, wenn folgende Verknüpfung zwischen den Bedingungen A und B bestehen:

- A ist wahr und/oder B ist wahr;
- A ist falsch und/oder B ist wahr;
- A ist wahr und/oder B ist falsch;
- A ist falsch und/oder B ist falsch.

Weitaus kompliziertere Verknüpfungen sind möglich:

- A falsch und B wahr oder A wahr und B falsch;
- A wahr und B wahr oder A falsch und B falsch.

Anforderungen dieser Art werden im kommerziell-betriebswirtschaftlichen Bereich, mit dem wir uns in diesem Buch besonders befassen, kaum auftreten.

Mit diesen Andeutungen über besondere Nutzungsmöglichkeiten von PROTARen möchte ich es hier bewenden lassen.

4.7.6. Rechenzeit

Man mag meinen, daß dem Problem der Rechenzeit bei der Unterweisung von Anfängern im Programmieren von PROTARen zu große Bedeutung beigemessen wird. Das trifft aber

nur teilweise zu. Denn der ernsthafte Programmierer, der seine Programme auch selbst benutzt, wird nämlich sehr bald auf die feinen Unterschiede in den Rechenzeiten aufmerksam, die er durch besondere Programmgestaltung sehr wesentlich beeinflussen kann. Denn mit wachsender Übung wird er seinen PROTAR immer häufiger bis an die Grenzen der Programmspeicherkapazität ausnutzen. Er wird daher jede Einsparung an Programmspeicherplätzen begrüßen. Im wesentlichen betrifft dies das Problem der Plazierung der Unterprogramme, das einen wesentlichen Einfluß auf die Rechenzeit ausübt.

Eines ist sicher, jedes Anspringen eines Unterprogrammes kostet mehr Zeit, als wenn der in einem Unterprogramm verpackte Programmzweig als Teil des Hauptprogrammes deklariert worden wäre. Solange noch genügend Programmspeicherplätze vorhanden sind, wird man davon absehen, mehrere untereinander völlig identische Programmzweige über ein Unterprogramm aufzurufen. Im Falle eines Mangels an Programmspeicherplätzen ist abzuwägen, welche der beiden Forderungen, Vorrang hat, die nach Realisierung zusätzlicher Informationswünsche oder die nach erhöhter Rechengeschwindigkeit. Im Rahmen der PROTARe dürfte die Frage der Rechenzeit im allgemeinen nicht so sehr ins Gewicht fallen. Der Wunsch nach größerer Geschwindigkeit ist eher psychologisch bedingt. Es wird aber sicherlich auch Fälle geben, in denen eine erhöhte Rechengeschwindigkeit sicherlich ein Argument sein wird. Dies wird vor allem dann zutreffen, wenn Sie beispielsweise eine größere Anzahl automatisch ablaufender Kalkulationen zu bewältigen haben. Diese Technik wird im Beispiel B im Anhang behandelt.

Die Stellen, an die man Unterprogramme vorteilhaft setzt, werden in erster Linie vom Rechner bestimmt. Im Abschnitt 3.3. über die Programm-Marken haben wir gehört, in welcher Art und Weise verschiedene Rechner das Suchen eines Labels durchführen. Daraus läßt sich auch die wirtschaftlichste Plazierung von Unterprogrammen ableiten. Denn der kürzeste Weg ist auch für die Rechenzeit der günstigste.

Bei den Rechnern, die ab dem augenblicklichen Stand des Programmzeigers die Anweisungen zyklisch nach dem Label des Unterprogrammes absuchen, wird das Unterprogramm am besten gleich nach dem Hauptprogramm zu setzen sein. Programme für Rechner, die mit dem Suchen des Labels aber grundsätzlich am Programmanfang beginnen, erhalten die Unterprogramme an den Programmanfang gesetzt. Dabei wird man selbstredend von mehreren Unterprogrammen jene ganz an den Anfang stellen, die am häufigsten vom Hauptprogramm aus aufgerufen werden.

In den Bedienungsanleitungen sind die Vorgänge um das Aufsuchen des Labels nachzulesen. Läßt Sie diese Anleitung doch einmal im Stich, wird empfohlen, vom Hersteller eine detaillierte Beschreibung dieser Vorgänge anzufordern.

Zusammenfassend zur Rechenzeit sei wiederholt, daß Unterprogramme nur dann gesetzt werden sollen, wenn dies aufgrund des Mangels an Programmspeicherplätzen erforderlich ist. Dann ist die Priorität der Forderungen nach kürzerer Rechenzeit oder zusätzlichen Informationen festzulegen, nach der sich die Gestaltung des Programmes richtet.

5. Formblattwesen

Wozu im Zusammenhang mit PROTARen überhaupt über Formblätter reden? Rechner und Formblätter sind doch zwei verschiedene Paare Schuhe, richtig! Bei näherem Hinsehen sind aber doch Zusammenhänge zu erkennen. Und diese erscheinen mir als so wesentlich, daß über diese im Rahmen der vorliegenden Erörterungen ebenfalls zu diskutieren ist.

Formblätter sind die Zwangsjacken der Bürotechnik und des täglichen Lebens. Wie oft haben Sie sich schon beim Ausfüllen von Vordrucken darüber geärgert, daß Sie für bestimmte Eintragungen zu wenig und für andere zuviel Platz hatten, möglicherweise stand der zur Verfügung stehende Raum jeweils im umgekehrten Verhältnis zur Wichtigkeit der Eintragung! Dabei dachten Sie sicherlich auch wie ich, daß der „Erfinder" solcher Formblätter seine Produkte zur Strafe einen Monat lang täglich acht Stunden lang selbst ausfüllen sollte!

Diese kritischen Bemerkungen gelten im besonderen Maße für die DV und daher sind folgende grundsätzliche, vorerst nicht nur PROTARe betreffende Bemerkungen diesem Kapitel voranzustellen:

1. Formblätter, die im Geschäftsverkehr nach außen gelangen, werden im allgemeinen von den Herstellern der Rechenanlagen mitgeliefert. Zumindest werden Vorschläge über die Gestaltung der Vordrucke dem Käufer unterbreitet.

2. Formblätter für den internen Bereich gibt es im allgemeinen nicht; Statistiken werden meist auf Zeilenfinderformularen ausgedruckt und die darauf erstellten Zahlenfriedhöfe erhalten bestenfalls Kennziffern. Nach diesen Kennziffern ist in einer an anderer Stelle befindlichen Legende nachzusehen, worauf diese Kennziffern hinweisen. Bei monatlich zu erstellenden Statistiken ist man nicht in der Lage, mehr als die wichtigsten Kennziffern und ihre Bedeutung im Kopf zu behalten. Daraus ergibt sich, daß DV-Statistiken so ungern gelesen werden, weil das Lesen viel zu anstrengend ist.

Die Ursachen der in Punkt 2 geschilderten Misere liegen meines Erachtens auf der Hand: Erstens wird im allgemeinen mit relativ kleinen und daher billigen Rechenanlagen gearbeitet, sodaß es schon die geringe Speicherkapazität verbietet, erklärende Texte in ein Programm aufzunehmen. Damit zusammenhängend ist für einen Kauf sicher der Preis ausschlaggebend. Andererseits würde auch der Zeitaufwand beim Programmieren größer, wenn man diese Texte einplant, was sich natürlich auch in den Programmkosten niederschlägt. So wird denn in erster Linie aus Kostengründen – auch das Ausdrucken der Texte kostet Zeit! – auf eine bessere Lesbarmachung der Statistiken verzichtet; dieser Umstand wird aber dem Kunden vorerst verschwiegen, weil sich dieser zum Zeitpunkt des Kaufes einer Rechenanlage über derartige Details meist noch gar keine Gedanken macht! Überlegungen der geschilderten Art können aber beim Kauf des zweiten Rechners mitentscheidend sein.

Daraus kann man die Konsequenz ziehen, Formblätter zu entwickeln, die Statistiken für die Geschäftsführung und andere nicht an deren Erstellung Beteiligte erst lesbar machen. Eine Vorlage derselben ohne erläuternde Bemerkungen würde zu zeitraubenden Rückfragen führen.

Die bisherigen Bemerkungen galten in erster Linie der **Ausgabe** verschiedener Ergebnisse. Der gleiche Grundsatz muß aber auch für die **Eingabe** gelten. In dem Fall genießt nicht nur der daraus einen Vorteil größerer Sicherheit, der am Rechner sitzt und ihn bedient, auch für die Erfassung der Daten bietet ein Formblatt die Gewähr für gesteigerte Zuverlässigkeit. Maschinenzeit kostet Geld. Überdies müssen andere auf die nächste freie Maschinenzeit länger als geplant warten, wenn durch Unklarheiten im Eingabebeleg Verzögerungen in der Verarbeitung auftreten.

Hier wird auch im großen Bereich der DV noch sehr viel verabsäumt. Eingaben sollten tunlichst übersichtlich in der Reihenfolge auf einem Eingabebeleg notiert sein, in der sie dann auch vom Rechner verarbeitet werden. Jedes Springen mit den Augen und das Suchen der nächsten Eingabe kostet nicht nur Zeit und damit Geld, sondern strengt auch sehr an. Ein echter Vergleich von Preis-Leistungsverhältnissen sind auch zum Zeitpunkt der Angeboterstellung und des Kaufes eines Rechners praktisch undurchführbar, wenn man dabei auch organisatorische Fragen mit einbeziehen will.

Damit sei auch wieder der Anschluß an unsere Probleme im „Zusammenleben" mit PROTAR gefunden: Unsere Rechner unterscheiden sich rein organisatorisch in nichts von ihren größeren Artgenossen. Hier wie dort wird auf festen Geleisen vorgegangen, die nicht verlassen werden können. Die Hinweise für die Eingaben können Wort für Wort übernommen werden. Auch PROTARe schätzen es, wenn man die Eingaben in immer der gleichen Reihenfolge macht und der Benutzer von PROTAR zieht es ebenso wie der Operator an einer größeren Anlage vor, mit den Augen nicht „springen" zu müssen.

In der Ausgabe allerdings können wir uns bisher nur im Fall eines Rechners (s. S. 120/6) mit einer Textausgabe helfen; allerdings wird dadurch der verwendete Papierstreifen auch nicht breiter. Formblätter ausdrucken können wir nicht. Aber die Ausgaben mit einem Index versehen und den Papierstreifen in ein Formblatt einkleben, das können wir. Es bleibt uns daher nichts übrig, wenn wir mit PROTARen erfolgreich zusammenarbeiten wollen, auch die Gestaltung der Formblätter für Ein- und Ausgaben entscheidend mitzubestimmen. Wenn Sie nun ein Formblatt entwerfen wollen oder müssen, dann ist zu empfehlen, ähnlich vorzugehen, wie bei der Analyse des Problems selbst. Auch hier müssen Sie darüber im klaren sein, was Sie wollen, woher Sie wie welche Unterlagen nehmen und wo Sie Ergebnisse hintun. Sie sehen die Reihenfolge der Eingaben auf den Formblättern so vor, wie Sie diese im Programm festgelegt haben. Das gilt sinngemäß auch für die Ausgaben, sodaß Sie nicht lange nach dem nächsten Ergebnisfeld auf dem Formblatt suchen müssen. Außer auf die zweckentsprechende Einteilung kommt es bei einem Formblatt auch auf eine optisch ansprechende, durch unterschiedliche Liniendicken differenzierende Gestaltung an. Am besten zeichnet man das Liniennetz in Originalgröße auf Millimeterpapier vor, experimentiert damit, ob Zeilenabstände und Spaltenbreiten den in der Praxis anfallenden Aufgaben gerecht werden und bespricht dann die endgültige Lösung mit einem geübten Zeichner oder Graphiker.

Ein Formblatt sollte also

- eine übersichtliche Gliederung von Ein- und Ausgaben aufweisen;
- Kommentare zu den Daten selbst enthalten. Standardmäßige Informationen können dabei bereits auf der Drucksorte vorgegeben werden;
- eine ablauforientierte Datenordnung haben;
- Titel und Datum erkennen lassen;
- auf den jeweiligen Sachbearbeiter hinweisen und
- für alle verständlich abgefaßt sein, damit man dessen Sinn erkennen kann.

Gerade die geringe Größe von PROTAR bewirkt, daß mitunter wegen seiner Unscheinbarkeit wenig auf seine Belange Rücksicht genommen wird. Daher sei hier erinnert, daß speziell bei der Bewältigung kaufmännisch-betriebswirtschaftlicher Probleme mit besonderem Nachdruck an die Gestaltung der Unterlagen gegangen wird. Je gewichtiger ein Problem ist, desto notwendiger wird auch dessen organisatorische Erfassung. Je kleiner es ist, desto eher kann man seine Anforderungen vermindern; aber gänzlich sollte die Mitgestaltung der Vordrucke nicht vernachlässigt werden, wenn man sich nicht selbst um einen Teil der Nutzungsmöglichkeiten seines PROTARes bringen will. Denn nur bei Berücksichtigung dieser organisatorischen Grundsätze kann auch PROTAR erst seine Fähigkeiten voll entfalten!

6. Anwendungen

Aufbauend auf den Unterweisungen in der Technik des Programmierens werden nun praktische Hinweise zu verschiedenen Anwendungsmöglichkeiten von PROTARen gegeben. Zunächst stellt sich die Frage, woher die Ideen, bzw. Anstöße zur Erarbeitung von Programmen kommen. Darauf gibt es meines Erachtens nur zwei Antworten:

1. Durch gezieltes Nachdenken wird geprüft, welche der anstehenden Probleme am besten für die Bearbeitung durch PROTAR geeignet sind.

2. Durch intuitives Erfassen eines Problems während seiner konventionellen Bearbeitung aufgrund eines augenblicklichen Einfalls.

Aus meiner eigenen Praxis kann ich sagen, daß der zweite Fall mit wachsender Routine in der Handhabung von PROTAR immer häufiger wird. Denn die hier mehrfach wiederholte These, daß PROTAR zur Lösung kleinster bis mittlerer Probleme – siehe Anlagenbuchhaltung – ein geeignetes Hilfsmittel ist, zeigt, daß man meist am Beginn einer schon mehrfach konventionell von Hand, mit einem Rechenschieber, auf einem TAR oder PROTAR abgewickelten Verarbeitung einmal zu dem Schluß kommt „Schluß mit der Routinearbeit!" und dann ein kleines Programm zu schreiben beginnt. Der **Anstoß kommt somit meist aus einer direkten Verarbeitung heraus** und diese wird dann unterbrochen, um das Programm zu schreiben; mir ist das beispielsweise bei einer Auswertung von Zeitaufnahmen so ergangen. Eigenartigerweise hatte ich diese Auswertungen schon trotz Vorhandensein meines PROTARs noch konventionell vorgenommen, bis eines Tages der berühmte „Funke" übersprang. Das gleiche galt für die Aufrechnung von Tabellen von Schlüssel-Zahlen für die Kostenstellen-Umlage in der Betriebsabrechnung. Hier waren bisher Zeilen und Spalten konventionell aufgerechnet worden. Etwa 60 bis 80 solche Listen ließen plötzlich den Mut zu weiterer konventioneller Aufrechnung schwinden, denn bei dieser Art der Verarbeitung mußte eine Zahl immer zweimal zur Bildung der Zeilen- und der Spaltensumme eingetastet werden.

Neben der individuellen Erarbeitung von Programmen bieten die Hersteller vielfache Anwendungsmöglichkeiten im Rahmen der Nutzung von Standardprogrammen. Die häufigsten Gebiete sind Themen aus der Mathematik, Statistik, aus den verschiedensten Gebieten der Technik, im Bauwesen vor allem in der Statik, für die Programm-Pakete angeboten werden. Da und dort bieten auch andere Stellen solche Standard-Lösungen an. Gewisse Programme gibt es auch zur Lösung kaufmännischer und finanztechnischer Probleme. In letztem Fall gibt es sogar auch TARe, also nicht programmierbare Taschenrechner, die für die Finanztechnik wichtige Funktionen bereits fix eingebaut haben, sodaß eine Programmierung ohnehin entbehrlich ist. Ein weiteres Anwendungsgebiet scheint mir die Möglichkeit zu sein, das Programmieren auf diesen kleinen Geräten zunächst einmal prinzipiell zu lernen, um dann später auch auf größeren Anlagen die Zusammenhänge leichter und damit schneller erfassen zu können. Im kleinen Rahmen kann die Lösung einer Problematik geübt werden; die Ergebnisse dieser Übungen kommen dann später zum Tragen. Außerdem lassen sich Detailprobleme, wie das Einordnen von Zahlen der Größe nach schon auf dem kleinsten PROTAR durchführen. Auf ihm kann man kurzfristig ein Programm ändern oder völlig neu schreiben und zusammen mit dem Test ist ein Programmdetail in relativ kurzer Zeit gelöst und kann dann in ein Programm auf einem größeren Rechner eingebaut werden. Dadurch wird uns auch die Arbeit der Analyse durch Teilung der Probleme wesentlich erleichtert.

Wozu ist also PROTAR verwendbar?

1. Vorprogrammierung und Test von Detailproblemen, die dann auf größeren Rechenanlagen realisiert werden.

2. Für selbständige Verarbeitung nach speziell für PROTAR geschriebenen Programmen und

3. als konventionelles Rechengerät.

6.1. Einsatzmöglichkeiten von Kleinrechnern

Wer braucht einen programmierbaren Kleinrechner? Prinzipiell sind alle jene zu möglichen Anwendern von PROTARen zu zählen, die schon immer Rechenhilfen benutzt und dabei immer wiederkehrende Probleme zu bearbeiten hatten. In erster Linie sind dies Betriebe, Prüfanstalten, Ingenieurbüros und wissenschaftliche Institute.

In den Betrieben sind es vorrangig die Abteilungen, die sich mit Kostenrechnung und Kalkulation befassen, in denen man aus einer beschleunigten Abwicklung immer wiederkehrender Rechenvorgänge großen Nutzen zieht. Auch im Verkauf können sich Ansatzpunkte für Anwendungen von PROTAR bieten.

Prüfanstalten, die mit der Güteprüfung von Rohmaterialien und Fertigprodukten befaßt sind, haben Ergebnisse gleichartig ablaufender Prüfvorgänge auszuwerten.

Ingenieurbüros benötigen für die Konstruktion und Berechnung von Projekten in erster Linie die Ermittlung der Auswirkungen verschiedener Beanspruchungen der Bauteile. Hier helfen vorrangig Programme aus der Statik.

In der Forschung wird man es dagegen mit weniger häufigen, aber um so rechenintensiveren Problemen zu tun haben. PROTARe sind daher auch in wissenschaftlichen Instituten willkommen.

6.2. PROTAR und Großrechenanlage

Nehmen wir an, Sie wären Konstrukteur und in Ihrem Betrieb wäre eine Großrechenanlage installiert. Für ein dringendes Angebot würden Sie für die Lösung eines statischen Problems, bei dem Sie mehrere Belastungsfälle durchrechnen wollen, um die kostengünstigste Konstruktion herauszufinden, eine kurze Rechenzeit benötigen. Nun bekommen Sie aber keine Rechenzeit. Ihr Kunde wird wenig Verständnis für die mangelnde Koordination der auf Ihrer Großrechenanlage abzuwickelnden Aufgaben haben, wenn Sie ihn wegen des Angebotes auf den nächsten Tag vertrösten. Einen möglichen Auftrag sind sie mit Sicherheit los. Eine Ausstattung der Arbeitsplätze mit Bildschirmgeräten, die an die Großrechenanlage angeschlossen werden, war aus Kostengründen in Ihrem Betrieb unterblieben. Mit PROTARen kann diesem Mangel zum Teil abgeholfen werden. Eine Ausrüstung der Arbeitsplätze mit PROTARen, um Computerleistung auch dort zur Verfügung zu haben, wäre in diesem Fall die beste Lösung. PROTAR und Großrechenanlage können sich somit ergänzen und stehen absolut in keinem Gegensatz zueinander.

6.3. Organisation einer Programmierung

Die Lösung dieser Frage steht in engem Zusammenhang mit der Ausstattung eines Betriebes oder eines Institutes mit Kleinrechnern. Werden PROTARe nur vereinzelt angewendet und steht vor allem nur ein PROTAR in einer Abteilung im Einsatz, wird jeder Benützer eines PROTARes mit der Technik der Programmierung voll vertraut sein müssen. Ist aber eine Abteilung mit mehr als einem Gerät ausgestattet, kann an eine zentrale Programmierung in dieser Abteilung gedacht werden. Für die Lösung von untergeordneten Problemen sollte trotzdem jeder Benützer eines Kleinrechners programmtechnisch geschult sein.

Die Programmierung von PROTARen außer Haus ist nicht die erstrebenswerteste Lösung unseres Programmbedarfes. Wenn wir nicht Standardprogramme des Herstellers unseres PROTARs und Programme aus Programmbibliotheken verwenden, werden wir uns der **Eigenprogrammierung** zuwenden. Gerade die Kleinheit unseres Rechners gestattet es uns an individuellen Lösungen selbst zu arbeiten. Einerseits kennen wir den Rechner in allen seinen Funktionen und andererseits kennen nur wir unsere eigenen Probleme wirklich. Es wäre auch unwirtschaftlich, einem Außenstehenden die Umstände zu erklären, warum wir eine ganz bestimmte Vorgangsweise wählen, wo er unter Umständen das Problem ganz anders gelöst hätte.

Da sind es gerade die im kommerziell-betriebswirtschaftlichen Bereiche Tätigen, die praktisch nur von kleinen individuell erstellten Programmen leben.

Über die Bedienungsanleitung hinaus ist vom Hersteller unseres PROTARs keine Unterstützung in der Programmierung zu erwarten. Dieser Umstand darf nicht nur negativ gesehen werden. Denn die Preiswürdigkeit der Rechner hat in diesem Verhalten der Lieferfirmen eine Ursache. Mit dem in den Anleitungen und allfälligen Kundendienstzeitschriften angebotenen know-how und der eigenen Praxis kann es durchaus zu verblüffenden Problemlösungen kommen.

6.4. Anwendungsprogramme

6.4.1. Standardprogramme

Die Hersteller von PROTARen bieten Standardprogramme für die verschiedensten Gebiete, wie Mathematik, Statistik und einige Bereiche der Technik an. An sich ist nichts gegen die Anwendung solcher Programme zu sagen. Wer in einem entsprechenden Fachgebiet angesiedelt ist, für das Standardprogramme am Markt sind, sollte sie ruhig verwenden. Er wird sicherlich hie und da Problemlösungen finden, die für seine Arbeit eine große Hilfe darstellen. Dazu sollte man sich von den Herstellern die Kurzbeschreibungen jener Programme kommen lassen, deren Anwendung ins Auge gefaßt wird. Dabei wird man sehen, was das Programm kann und ob es tatsächlich für den beabsichtigten Verwendungszweck anwendbar ist. Dabei muß man darauf gefaßt sein, daß fast alle Angaben in englischer Sprache abgefaßt sind.

Man wird sich bei der Verwendung von Standardprogrammen oder Programmen aus Programmbibliotheken mit einer Tatsache auseinandersetzen müssen: Das Programm wurde aus dem Blickwinkel des Programmverfassers geschrieben. Das bedeutet, daß man sich voll und ganz der vom Autor des Programmes vorgesehenen Lösungsmethode anschließen muß. Wie weit dies auch möglich und wirtschaftlich ist, sollte vor dem Kauf des Programmes nach dem Studium der Programmbeschreibung festgestellt werden.

Vor allem Programme aus Programmbibliotheken sind von deren Verfassern in erster Linie zur Abdeckung individueller Bedürfnisse erstellt worden.

Früher oder später wird der Benutzer doch den Vorteil von PROTAR erkannt haben, den der verhältnismäßig einfachen Eigenprogrammierung.

6.4.2. Individualprogramme

Im Verlaufe unserer Praxis haben wir Probleme zu formulieren gelernt. Aus der Sicht der Programmanwendung haben wir auf die optimale Gestaltung unserer Problemlösungen wesentlichen Einfluß genommen. In der Verarbeitung ist exakte Vorbereitung zur Gewohnheit geworden. Diese Eigenschaften befähigen nach Ansicht des Verfassers zur Erstellung funktionsfähiger Individualprogramme.

Zur Demonstration der Möglichkeiten individueller Programmgestaltung sei hier bereits auf den Beispielteil im Anhang verwiesen. Einige der nachfolgend angeführten Problemlösungen werden dort näher erläutert. Die Erwähnung von Programmen erhebt keinerlei Anspruch auf Vollständigkeit und ist in erster Linie als Anregung gedacht. Der Leser kann sich daraus ein Bild machen, was er für seinen Bedarf an Nutzen aus PROTAR herausholen kann.

Beispiele aus dem innerbetrieblichen Rechnungswesen

- Kalkulation von Artikeln eigener Erzeugung mit manueller Eingabe der Stammdaten (Beispiel A).
- wie vor, jedoch mit automatischer Zuführung der Stammdaten über Datenprogramme (Beispiel B).
- Mengenauflösung mit Hilfe der bereits im vorigen Beispiel verwendeten Datenprogramme zur Ermittlung von Kostenartensummen (Beispiel C).
- Anlagenbuchhaltung zur Errechnung der kalkulatorischen Kosten, wie Abschreibungen und Zinsen, mit manueller Eingabe der Daten (Beispiel D).
- wie vor, jedoch mit automatischer Zuführung der Daten (Beispiel E).
- Auftragsbearbeitung (Beispiel F).
- Energieverbrauchskontrollen (Beispiel G).
- Kalkulation von Zeitaufwendungen für verschiedene Arbeitsgänge in der Produktion aufgrund von Zeitaufnahmen, anstelle der Anwendung von Normzeittabellen.
- Auswertung von Zeitaufnahmen (Beispiel H).
- Aufrechnung von Tabellen nach Zeilen- und Spaltensummen zur Vermeidung zweimaliger Zahleneingaben (Beispiel I komplett mit Programmunterlagen).

Beispiele aus der Prüftechnik

- Speicherung von Rezepturen zur Ermittlung artikelbezogener Anteile der einzelnen Komponenten.
- Statistische Erfassung von Materialkennwerten.
- Optimierung eines Zuschlagstoffgemenges aus maximal vier Korngruppen (geeignet zur Ermittlung der günstigsten Zusammensetzung für die Betonerzeugung).
- Auswertung einer Siebanalyse von Zuschlagstoffen für die Betonherstellung gemäß ONORM B 3304.
- Güteprüfung von Produkten eigener Erzeugung. Im Beispiel K wird diese an einem Produkt der Betonsteinindustrie, dem Bordstein, demonstriert.

Diese Liste ließe sich beliebig vergrößern, bezieht sich hier auf bestehende Verwirklichungen von Programmen. Im Bedarfsfall kann der interessierte Leser über briefliche Anfrage beim Autor Unterlagen über die hier angeführten Programme erwerben.

Im übrigen wird darauf hingewiesen, daß weitere Bände aus der gleichen Buchreihe in Vorbereitung sind und demnächst erscheinen werden. Sie befassen sich ausführlicher mit verschiedenen Anwendungen. Es sind dies Bücher für folgende Bereiche:

- Kalkulation in Produktionsbetrieben,
- Auswertung von Baustoffprüfungen und
- Ermittlung von kalkulatorischen Abschreibungen und Zinsen von Anlagegütern.

7. Auswahlkriterien

Vor der Anschaffung eines programmierbaren Taschenrechners muß man sich darüber im klaren sein, zur Lösung welcher Probleme man ihn einzusetzen gedenkt. Leider ist es verhältnismäßig schwierig, beim Kauf von TAR und PROTAR eine wirklich sachkundige Beratung zu erhalten. Schon für Fachhändler ist es nicht leicht, sich in die unterschiedlichen Bedürfnisse einzelner Interessenten hineinzudenken und diese gezielt zu beraten. Soweit aber die Rechner über Kaufhäuser oder Ladenketten angeboten werden, ist von diesen nur in Ausnahmefällen ein über das allgemein Grundlegende hinausgehender Beratungsdienst einkalkuliert. Auf diese Weise gelangen die Rechner wohl relativ billig an den Endverbraucher. Die zwangsläufige Begleiterscheinung ist aber, daß dieser nicht immer das für ihn geeignetste Gerät ausfindig macht. Daher sei im folgenden eine kleine Übersicht gegeben, nach der eine Auswahl wenn schon nicht in jedem Fall getroffen, so doch wesentlich erleichtert werden kann. Dabei sollten Sie sich wirklich hinsichtlich Ihrer Anforderungen an PROTAR im klaren sein.

Grundsätzlich sollte jeder PROTAR mit aufladbaren Batterien und Netzanschluß augestattet sein. Die Batterien werden auch während des Betriebes vom Netz her aufgeladen. Der Netzstecker des Anschlußkabels ist dabei als Ladegerät ausgebildet. Die Lebensdauer der Batterien ist relativ groß. Mit einer Ladung können Sie zwischen zwei und vier Stunden netzunabhängig rechnen. Die tatsächliche Zeitspanne hängt von so vielen Faktoren ab, daß genaue Angaben nicht gemacht werden können. Rechner mit laufender Anzeige während eines Programmlaufes verbrauchen mehr Energie als solche, deren Anzeige während einer Verarbeitung erlischt. Mit diesen aufladbaren Batterien sind Sie nicht an Ihren Arbeitsplatz gebunden. Sind Sie mit PROTAR unterwegs und beabsichtigen Sie ihn häufig zu benutzen, empfiehlt sich die Mitnahme einer Garnitur Ersatzbatterien. Die Aufladung einer Garnitur kann über Nacht im Hotel erfolgen.

Für manche PROTARe werden Zusatzeinrichtungen angeboten. Bei nicht sofortigem aber später geplanten Bedarf sollte mit dem Erwerb nur solange zugewartet werden, als das zum Kauf vorgesehene Modell normal im Handel erhältlich ist. Zu einem späteren Zeitpunkt ist es unsicher, ob die dann angebotenen Zusatzeinrichtungen noch auf die älteren Modelle passen.

Die nachfolgenden Angaben über Auswahlkriterien erheben in keiner Weise Anspruch auf Vollständigkeit. Denn einerseits ist die Entwicklung im Bau der PROTARe in keiner Weise abgeschlossen, andererseits mögen in manchen Fällen in Zukunft durchaus andere Argumente im Vordergrund stehen. Die nachfolgende Zusammenstellung dürfte aber die meisten Interessenten an PROTARen vor Fehlentscheidungen beim Kauf bewahren.

Problemstellung	Rechnerauswahl
Es besteht in der Hauptsache nur ein Problem	Spezifische Rechner ohne Programmierung mit verdrahteten Funktionen; Vorteil: rasche Rechenzeit, oder
	Rechner mit Konstantspeicher für Programme
Viele kleine Probleme	Rechner ohne Magnetstreifen mit maximal 50 bis 100 Programmschritten.
Die Ergebnisse müssen dokumentiert werden	Rechner mit Druckeinrichtung (Thermodrucker)
Viele Ausgaben laufen über die Anzeige	Wahl eines Rechners mit großen Zeichen in der Anzeige
In erster Linie größere Probleme	Rechner mit Magnetstreifen und großer Programmspeicherkapazität.
Unterschiedliche Probleme mit wechselndem Bedarf an Programm- und Datenspeichern	Rechner mit frei wählbarer Speicherteilung
Kaufmännische Probleme	Rechner mit Druckeinrichtung
Technische Probleme	Rechner mit größerer Rechengeschwindigkeit
Ein Programm wird nur einmal benutzt	Rechner ohne Druckeinrichtung
Programme, die für Außenstehende nicht eingesehen werden dürfen	Rechner mit Magnetstreifen und Sperre für Programm-Listung
Standardprogramme	Rechner mit Magnetstreifen, bzw. Programm-Modulen
Fortzuschreibende Ergebnisse wie bei Konten	Rechner mit Magnetstreifen
Rechner für rechenintensive Probleme	Rechner mit algebraischer Logik
Kaufmännische Probleme und Kalkulationen	Rechner mit %-Taste
Dauerbetrieb	Rechner mit größerem Tastenfeld und übersichtlicher Anzeige

Nachwort

Zunächst dringende Bitten an die Hersteller von PROTARen und zwar jeweils an die, die sie angehen: Vor dem Kauf eines Rechners sollte mehr und ausführlichere Information über die Möglichkeiten der Rechner an den Kunden gelangen. Auch sollten Service-Netze auf- und bestehende ausgebaut werden. Eine Reparatur eines PROTARs sollte innerhalb von 24 bis 48 Stunden durchgeführt werden können. Es geht nicht an, daß bei einer Reparatur mit einem Ausfall von zwei bis vier Wochen gerechnet werden muß! Sind in einem Betrieb mehrere PROTARe von einem Modell eingesetzt, kann man sich gegenseitig aushelfen. Sollte aber nur ein Gerät im Hause sein, werden wesentlich kürzere Reparaturzeiträume unentbehrlich. PROTARe sollten durch besser geschultes Personal verkauft werden als zur Zeit. Damit würden sich zugleich die Hersteller selbst einen Dienst erweisen, denn die weitere Verbreitung gerade qualifizierter, also teuerer Rechner würde durch bessere Beratung der Interessenten entscheidend gefördert werden.

Auch wenn sie nicht hauptberuflich Programmierer werden wollen, werden Sie erkannt haben, daß individuelle auf die eigenen Bedürfnisse zugeschnittene Programme oft wirtschaftlicher sind, als andere Lösungen, sodaß sich der Aufwand für die Durcharbeitung dieses Buches auf alle Fälle gelohnt hat.

Hinweise auf Rechnermodelle

Die genannten Modelle sind im allgemeinen überall erhältlich, sodaß hier auf die Anschriften der Lieferfirmen verzichtet werden kann.

1. Fa. HEWLETT-PACKARD;
2. FA. TEXAS INSTRUMENTS;
3. Modelle HP 67 und HP 97 der Fa. HEWLETT-PACKARD;
4. Modelle HP 33 E und HP 38 E der Fa. HEWLETT-PACKARD;
5. Modelle HP 25 C und HP 29 C mit „Constant Memory" der Fa. HEWLETT-PACKARD;
6. Modell TI 59 der Fa. TEXAS INSTRUMENTS;
7. Modelle TI 58 und TI 59 der Fa. TEXAS INSTRUMENTS;

Beispielsammlung

Vorbemerkung:

Es würde den Rahmen des hier Dargestellten sprengen und ihn wesentlich überschreiten, würde bei diesen Beispielen auf einzelne Programmdetails eingegangen werden; da und dort wo es für das Verständnis der Zusammenhänge wesentlich erscheint, wird man das tun, aber nicht lückenlos durch den gesamten Fall durchziehen. Dazu kommt auch noch, daß bedingt durch die Individualität der Lösungen auch die Grundlagen, die zu einem Programm geführt haben, von Ort zu Ort verschieden sein können.

Hier soll die Möglichkeit geboten werden, sich noch intensiver mit einem Problem selbst zu befassen, wie das während der Erörterung der zum Teil doch theoretischen Grundlagen nicht so möglich war, um nicht zu sehr vom Thema abzuweichen. Die Beispielsammlung, ausführlich und komplett dargestellt, würde eine eigene Broschüre füllen können. Wir müssen hier versuchen, uns auf das wesentliche zu beschränken, ohne daß das Verständnis zu sehr darunter leiden muß. In einem Fall wird dem Leser ein komplettes Programm demonstriert (Beispiel I).

Beispiel A: **Industrielle Kalkulation**

START	VERR.KS	VERLUST	STORNO SOKO	STORNO ST
ART-Nr	II.E	DSP II.E	SOKO	STAMMD

An die Spitze haben wir ein Abbild des Magnetstreifens gestellt, der auf unserem Rechner zusätzlich zu seiner Funktion als Programmspeicher die Bedienerführung übernimmt. Aus der Besprechung der den einzelnen Funktionstasten zugeordneten Funktionen wird man mit dem Programm näher bekannt.

START: Bei Beginn der ersten Verarbeitung nach Einlesen des Programmes ist das Programm zu starten; nach Beendigung des ersten Beispieles erfolgt dieser Start automatisch. Dabei werden nur die Speicher für die Grenz- und Vollkostensummen gelöscht und der Index auf den ersten möglichen Kostenartenschlüssel 2.1 eingestellt, der dann auch in der Anzeige erscheint.

VERR.KS betrifft die Eingabe einer Kostenstelle, bzw. die Eingabe des Gemeinkostenverrechnungssatzes und des dazugehörigen Variators, soweit diese nicht auf dem Datenstreifen gespeichert sind. Dabei wird das alte Wertepaar überschrieben. Der Vorgang betrifft jene Kostenstelle, auf der das zu kalkulierende Produkt erzeugt wird. Da die Datenspeicherkapazität nicht ausreicht, sämtliche notwendigen Kostenstellen zu speichern, werden die wechselnden Kostenstellen, das sind jene der Erzeugung, als variable Kostenstellen geführt. Im angeschlossenen Formblattbeispiel werden wir diesen Programmzweig nicht benötigen, da alle vorhandenen Produktionskostenstellen untergebracht werden konnten.

VERLUST: Bei jeder Produktion ist mit einem mehr oder minder großen Ausschuß zu rechnen. Dabei sind verschiedene Artikelgruppen von unterschiedlichen %-Sätzen betroffen. Über diese Taste wird der Verlustprozentsatz eingegeben und der alte überschrieben.

STORNO SOKO: Jede Eingabe von Sonderkosten sollte rückgängig gemacht werden können, um Eingabefehler unmittelbar – dies ist Voraussetzung – nach einer Eingabe korrigieren zu können.

STORNO ST: Auch die Stammdateneingabe muß nicht immer frei von Irrtümern sein. Im Zweifelsfall sollte nach der Stammdateneingabe dieselbe storniert werden. Auch hier muß die Korrektur unmittelbar nach der fehlerhaften Eingabe erfolgen.

ART-NR: Da wir nur einen schmalen Kontrollstreifen haben, werden wir einer Preisangabe den Andruck einer Artikel-Nummer voranstellen. Sie können auch andere Daten über diese Taste eingeben und andrucken lassen, beispielsweise ein Datum – allerdings ohne Zwischenräume und Sonderzeichen! – oder eine Kalkulations-Nummer.

II.E: Wir müssen unsere Kostenartengliederung so treffen, daß wir darin alle unsere Produkte einordnen können. Nicht alle Kostenarten werden im Verlauf einer Kalkulation benötigt. Wir müssen also Kostenarten überspringen können. Das bewerkstelligen wir über die Taste B, indem wir den nächsten gewünschten Kostenartenschlüssel eingeben. Der Rechner ist dabei angewiesen, unzulässige Schlüssel abzulehnen und dabei den jeweils letzten Index wieder anzuzeigen. Bei Verarbeitung einer Kostenart wird nämlich anschließend automatisch die nächste mögliche Kostenart angezeigt.

DSP II.E: Zwischenrechnungen sind oft unvermeidlich. Dabei vergißt man mitunter, welche Kostenart als nächste zur Eingabe vorgesehen war. Durch Drücken dieser Taste wird dieser Index oder Kostenartenschlüssel angezeigt.

II geht von 2 bis 19 und bezeichnet die Adresse des Datenspeichers (Speicher 0 und 1 werden für die Aufsummierung der Kosten verwendet).

E bezeichnet den Speicherteilindex und besagt, welchen Teil wir für die Verrechnung benötigen, den linken Teil des Speicherinhaltes vom Dezimalpunkt aus gesehen (1) oder den rechten (2).
Wie wir schon gehört haben, werden je Speicher zwei Verrechnungssätze mit je einem Variator gespeichert; die Darstellung des Speicherinhaltes sieht wie folgt aus:

SSSVV.SSSVV
linker Teil (1) rechter Teil (2)

SSS bezeichnet den Verrechnungssatz und VV den Variator – Anteil der variablen Kosten in ganzen Prozenten vom Verrechnungssatz. Die jeweilige Mengeneinheit eines Kostenarten-Verrechnungssatzes ist auf jene Größe abgestimmt, die wegen der Speichertechnik erforderlich ist.

Zum Beispiel wäre ein Kostensatz mit

2,40

gegeben. Ein Dezimalpunkt kann dabei nicht berücksichtigt werden, weil wir den ja zu der Darstellung der Speicherinhalte benötigen. Wir wählen daher die Mengeneinheit mit 100 und erhalten den Kostensatz mit

240.

Die artikelbezogene Mengenangabe muß dies natürlich berücksichtigen. Wäre diese Menge mit

5

gegeben, muß der Ansatz nun mit

0.05

getroffen werden. Dies gleich hier zum besseren Verständnis; wir bringen ja in einem Kostensatz nur drei wertführende Ziffern unter.

SOKO: bewerkstelligt die direkte Eingabe von Sonderkosten dem Wert nach getrennt nach Grenz- und Vollkosten in die Summenspeicher.

STAMMD: Mit dieser Taste wird jeweils die Menge der angesprochenen Kostenart eingegeben. Diese Menge ist auf die Größe des gespeicherten Kostensatzes abgestimmt, wie schon erläutert wurde.

Zu diesem Programm gehört auch ein Ladeprogramm, um die Verrechnungssätze mit deren Variatoren eingeben und abspeichern zu können. Dabei ist es auch möglich, nur einzelne Sätze allein anzusprechen und ihre Größen zu ändern. Die **Mengendaten** sind auf einem Artikel-Stammblatt aufzuzeichnen, sodaß jederzeit nach dem letzten Stand der Verrechnungssätze eine Kalkulation binnen kürzester Zeit ausgeführt werden kann. Nach Eingabe der letzten Menge erfolgt Andruck der Gesamtkosten nach Grenz- und Vollkosten mit Variator; zuvor wurden auch schon die Herstellkosten in der gleichen Art ausgedruckt.

Die Vorgangsweise bei Benutzung des Programmes ist folgende:

1. Einlesen der Programmkarte.
2. Einlesen der Datenkarte mit den Verrechnungssätzen.
3. Programm starten.
4. Eingabe Artikel-Nummer, evtl. auch Datum und Kalkulations-Nummer.
5. laufende Mengeneingaben vom Artikel-Stammblatt.

Ein Artikel-Stammblatt und ein Verrechnungsdatenblatt werden mit angeschlossen.

Die angeführte Kostenartenstruktur entspricht der eines Betonwerkes. Die gleichen Angaben finden sich auch auf dem im Beispiel B angeschlossenen Formblatt Datenprogramm-Generator wieder. Diese Methode eignet sich besonders für die Kalkulation von Sonderaufträgen; auch Standardartikel lassen sich damit kalkulieren. Wenn je Artikel nicht mehr als etwa 20 bis 25 Kostenarten vorliegen, sind Standardartikel noch vorteilhafter mit der im Beispiel B dargestellten Methode zu rechnen.

Der Vollständigkeit halber sollte das verwendete Kalkulationsschema skizziert werden:

	Materialkosten Löhne Fertigungsgemeinkosten
Andruck:	Herstellkosten (automatisch) Verwaltungs- und Vertriebskosten (z. T. automatisch)
	Selbstkosten Gewinn und Risiko (automatisch)
Andruck:	Gesamtkosten netto ohne MwSt. (automatisch)

Erläuterungen zu den Kostenarten

2.1 und 2.2 bezeichnen die Kosten von Beton-Mischanlagen.

3.1 bis 4.2 vertreten verschiedene Sorten an Zementen.

5.1 bis 8.2 stehen für verschiedene Qualitäten von Betonschotter.

9.1 bis 10.2 sind gesetzt für einzelne Baustahlqualitäten.

11.1 bis 12.1 werden den Produktionslöhnen verschiedener Erzeugungsbereiche zugeordnet.

12.2 bis 18.2 beinhalten die Gemeinkostenumlagen von Produktions- und sonstigen Kostenstellen, die bei der Erzeugung bis zum Versand durchlaufen werden.

19.1 und 19.2 werden für die Lohn- und Gemeinkostenanteile bei der Verladung eingesetzt.

Dipl. Ing. Gerfried Tatzl, WIV
Mörikestrasse 17,
A-8052 Graz-Wetzelsdorf

ARTIKEL-STAMMDATEN

Kalk.-Nr.
Datum

Artikel
Art-Nr
Volumen m^3
Menge
Gewicht kg
Bruch %

Schl.	Kostenart		Eht.	Menge/Einheit	Kosten
2.1	Mischerkosten Zentral-Mischanlage		ME		
2.2		Vorsatzbeton-Mischanlage	ME		
3.1	Bindemittel	PZ 275	t		
3.2		PZ 375	t		
4.1		PZ 475	t		
4.2		Weisszement	% kg		
	Sonderbindemittel	a 1)			
5.1	Zuschlagstoffe	Brechsand 0/2	t		
5.2		Splitt 2/5	t		
6.1		Splitt 5/8	t		
6.2		Sand 0/1	t		
7.1		Sand 1/4	t		
7.2		Kies 4/8	t		
8.1		Kies 8/16	t		
8.2		Marmorsplitt	t		
	Sonderzuschlagstoff	a 1)			
		a 1)			
9.1	Bewehrung	Rippentorstahl RT 50	% kg		
9.2		Rippentorstahl RT 60	% kg		
10.1		Betonrundstahl St 37	% kg		
10.2		Baustahlgitter	% kg		
	Sonderbewehrung	a 1)			
		a 1)			
11.1	Fertigungslöhne	Eisenbiegen und Erzeugen	h		
11.2	(Mann-h)	Nacharbeiten	h		
12.1		Transporte	h		
	Sonderkosten der Fertigung	1)			
		1)			
12.2	Gemeinkosten	Eisenbiegerei	h		
13.1	(Anlagen-h)	Bodenfertiger	h		
13.2		Plattenanlage	h		
14.1		Ringmaschine	h		
14.2		Rütteltisch	h		
15.1		Stampfbetonerzeugung	h		
15.2		Nacharbeiten	m^2		
16.1		Transporte	t		
16.2		Paletten	Pal.		
17.1		Produktionsboden	% m^2		
17.2		Dampfanlage	m^3		
18.1		Betrieb allgemein (11.1+11.2+12.1)	h		
18.2		Verwaltung und Vertrieb (automatisch)	%		
	Sonderkosten Verw. und Vertrieb	1)			
19.1	Löhne Verladung		h		
19.2	Gemeinkosten Verladung (20.1 Gewinn % automatisch)		t		

Bemerkungen 1) Sonderkosten mit Grenz- und Vollkosten max. 2-stellig
Vor neuer Kalkulation mit geändertem Bruch-% alten Satz durch Negativeingabe löschen !

ta 1978 06 02

Dipl. Ing Gerfried Tatzl, WIV Mörikestraße 17, A-8052 Graz - Wetzelsdorf	VERRECHNUNGSDATEN	gültig ab

Schl.	Kostenart		Eht.	Verrechnungssatz	Variator
2.1	Mischerkosten	Zentral - Mischanlage	ME		
2.2		Vorsatzbeton - Mischanlage	ME		
3.1	Bindemittel	PZ 275	t		
3.2		PZ 375	t		
4.1		PZ 475	t		
4.2		Weisszement	%kg		
5.1	Zuschlagstoffe	Brechsand 0/2	t		
5.2		Splitt 2/5	t		
6.1		Splitt 5/8	t		
6.2		Sand 0/1	t		
7.1		Sand 1/4	t		
7.2		Kies 4/8	t		
8.1		Kies 8/16	t		
8.2		Marmorsplitt	t		
9.1	Bewehrung	Rippentorstahl RT 50	%kg		
9.2		Rippentorstahl RT 60	%kg		
10.1		Betonrundstahl St 37	%kg		
10.2		Baustahlgitter	%kg		
11.1	Fertigungslöhne	Eisenbiegen und Erzeugen	h		
11.2	(Mann - h)	Nacharbeiten	h		
12.1		Transporte	h		
12.2	Gemeinkosten	Eisenbiegerei	h		
13.1	(Anlagen - h)	Bodenfertiger	h		
13.2		Plattenanlage	h		
14.1		Ringmaschine	h		
14.2		Rütteltisch	h		
15.1		Stampfbetonerzeugung	h		
15.2		Nacharbeiten	m^2		
16.1		Transporte	t		
16.2		Paletten	Pal.		
17.1		Produktionsboden	%m^2		
17.2		Dampfanlage	m^3		
18.1		Betrieb allgemein (11.1 + 11.2 + 12.1)	h		
18.2		Verwaltung und Vertrieb (automatisch)	%		
19.1	Löhne Verladung		h		
19.2	Gemeinkosten Verladung (20.1 Gewinn % automatisch)		t		

Beispiel B: **Industrielle Kalkulation automatisch**

Diese Methode wurde aus dem Beispiel A abgeleitet. Das Hantieren der Artikel-Datenprogramme ist durchaus mit der Hantierung von Magnetkonten vergleichbar. Größere Rechensysteme bieten für diesen Fall sicherlich elegantere Lösungen, aber zu einem wesentlich höheren Preis. Hier soll nur dargestellt werden, daß eine Automatik auch mit PROTAR zu erreichen ist. Wir bedienen uns dabei der gleichen Verrechnungsdaten, wie wir sie im Beispiel A kennengelernt hatten. Hier ist allerdings zu empfehlen, für jede Produktionskostenstelle eine eigene Datenkarte zu verwenden, auf der nur jeweils die benötigten Sätze gegenüber den anderen Datenkarten geändert sind, da im Programm eine gesonderte Eingabe von Verrechnungssatz und Variator nicht mehr vorgesehen ist. Dies gilt selbstverständlich auch nur für jene Fälle, in denen nicht alle Produktionskostenstellen unterzubringen sind und für die ein variabel gehaltener Speicher vorzusehen war. In unserem Beispiel war dieser Umstand nicht gegeben.

Das Programm selbst ist nur mehr das **Operationsprogramm**, das die eigentliche Kalkulation durch die Unterprogramme steuert und die Artikel-Stammdaten sind nun im restlichen Teil des Programmspeichers als Anweisungen enthalten; wir nannten diese Magnetstreifen **Daten-Programme**. Diese Streifen sind selbstredend mit den Artikel-Nummern zu beschriften. Zur Erstellung dieser Daten-Programme gibt es einen eigenen Programm-Generator. Die Artikelstammdaten werden in ein geändertes Stammblatt übertragen, das gleichzeitig Anweisungen gibt, wie die Daten nach Einlesen des Datenprogramm-Generators einzutasten sind. Die in unserem Beispiel verwendeten Kostenarten stimmen mit jenen in dem vorangegangenen Beispiel A in allen Punkten überein. Der Datenprogramm-Generator besteht im wesentlichen aus den Sprunganweisungen in die Unterprogramme und beinhaltet das Kalkulationsschema. Zwar könnte man die Datenprogramme für einzelne Artikel auch ohne Generator von Hand aus erstellen. Auf diese Weise könnten Programmspeicherplätze eingespart werden. Dieser Vorteil würde aber durch einen großen Nachteil erkauft werden: Dieser speicherplatzsparende Vorgang würde aber einen erheblich größeren Zeitaufwand beim Programmieren erfordern. Mit dem Programm-Generator ist man an ein Zeit sparendes Schema ein für alle Mal gebunden. Möglich ist die individuelle Gestaltung der Datenprogramme jedenfalls.

Die Kalkulation ist wie folgt beschrieben abzuwickeln:

1. Einlesen des Operationsprogrammes.
2. Einlesen der für die betroffene Kalkulation zugehörigen Datenkarte.
3. Programm starten.
4. Taste drücken, die die Zuordnung zweier Programme nebneinander im Programmspeicher ermöglicht.
5. Einlesen des Daten-Programmes.
6. Start der Verarbeitung.

Die Ausgaben sind die gleichen, wie bereits im Beispiel A beschrieben. Hier wird aber auch die jeweilige Artikel-Nummer automatisch angedruckt, weil sie im Daten-Programm mit gespeichert wird.

War die Verarbeitungsdauer im Beispiel A noch mit einigen Minuten pro Artikel zu bemessen, so beträgt hier die Verarbeitungsdauer nur mehr einen Bruchteil dessen und liegt bei etwa einer Minute.

Voraussetzung ist allerdings die Generierung der Datenprogramme und die laufende Pflege der Verrechnungssätze. Der Vollständigkeit halber wollen wir hier die Erstellung der Datenprogramme komplett darstellen:

1. Umschalten des Rechners in den Run-Modus.
2. Einlesen des Programm-Generators.

3. Umschalten auf Programmier-Modus.
4. Eintasten der Artikel-Stammdaten nach der auf den Stammblättern dargestellten Vorgangsweise.
5. Nach Eintasten des letzten Wertes Übernahme des Datenprogrammes auf Magnetstreifen und Beschriften desselben. Anschließend Rückkehr zu Punkt 1.

Diese Folge ist dann beliebig oft zu wiederholen. Am zeitaufwendigsten dabei sind allerdings die Vorarbeiten. Im allgemeinen stehen die Stammdaten wohl zur Verfügung, aber in einer Form, von der nicht immer direkt die Daten in unser Stammblatt übernommen werden können. Wir wollen ja auch die Sicherheit und das Vertrauen zu dieser Methode gewinnen. Zur Unterstützung rechnen wir ein paar Beispiele erstens einmal in gewohnter Weise durch, dann mit der Kalkulationsmethode nach Beispiel A und erst zum Schluß mit der soeben beschriebenen Methode; wir könnten uns ja bei der Erstellung des Programm-Generators oder des Operationsprogrammes geirrt haben. Der Fehler kann aber auch bei der Generierung des Daten-Programmes verursacht worden sein. Erst wenn diese Überprüfungen erfolgt sind, sollten wir mit der automatischen Kalkulation beginnen. An sich würde ich persönlich **jedes Daten-Programm** testen und zwar nach der Kalkulationsmethode nach Beispiel A. Das ist nicht sehr zweitaufwendig und bringt die Sicherheit für alle zukünftigen Anwendungen der Daten-Programme. Im letzten Satz wurden die Worte „zukünftige Anwendungen" gebraucht; daß diese nicht unbedingt nur die Kalkulation betreffen müssen, soll im nächsten Beispiel dargestellt werden.

Dipl. Ing. Gerfried Tatzl, WIV
Mörikestrasse 17,
A-8052 Graz - Wetzelsdorf

DATENPROGRAMM GENERATOR

PRODAT HP 97

Datum

Art-Nr

Artikel

Nr.	Kostenart		Eht	Schl.	Eingabe-Tasten 1)	GSB 9	SST-Tasten 3)		Anzeige 4) Zeile	Tastencode
1	Schalterstellung → RUN-Modus					2)				
2	Datenprogrammgenerator einlesen								0.	
3	Schalterstellung → PRGM-Modus								000	
4	Artikelnummer						SST			23 00
5						GSB				
6	Zentral-Mischanlage		ME	2.1		GSB	SST			23 01
7	Vorsatzbeton-Mischanlage		ME	2.2		GSB	SST			23 02
8	PZ 275		t	3.1		GSB	SST			23 01
9	PZ 375		t	3.2		GSB	SST			23 02
10	PZ 475		t	4.1		GSB	SST			23 01
11	Weisszement		%kg	4.2		GSB	SST			23 02
12	Brechsand 0/2		t	5.1		GSB	SST			23 01
13	Splitt 2/5		t	5.2		GSB	SST			23 02
14	Splitt 5/8		t	6.1		GSB	SST			23 01
15	Sand 0/1		t	6.2		GSB	SST			23 02
16	Sand 1/4		t	7.1		GSB	SST			23 01
17	Kies 4/8		t	7.2		GSB	SST			23 02
18	Kies 8/16		t	8.1		GSB	SST			23 01
19	Marmorsplitt		t	8.2		GSB	SST			23 02
20	Rippentorstahl RT 50		%kg	9.1		GSB	SST			23 01
21	Rippentorstahl RT 60		%kg	9.2		GSB	SST			23 02
22	Betonrundstahl St 37		%kg	10.1		GSB	SST			23 01
23	Baustahlgitter		%kg	10.2		GSB	SST			23 02
24	Löhne	Eisenbiegen u. Erzeugen	Mann-h	11.1		GSB	SST			23 01
25	Löhne	Nacharbeiten	Mann-h	11.2		GSB	SST			23 02
26	Löhne	Transporte	Mann-h	12.1		GSB	SST			23 01
27	Gemeinkosten	Eisenbiegerei	Anlagen-h	12.2		GSB	SST			23 02
28	Gemeinkosten	Bodenfertiger	Anlagen-h	13.1		GSB	SST			23 01
29	Gemeinkosten	Plattenanlage	Anlagen-h	13.2		GSB	SST			23 02
30	Gemeinkosten	Ringmaschine	Anlagen-h	14.1		GSB	SST			23 01
31	Gemeinkosten	Rütteltisch	Anlagen-h	14.2		GSB	SST			23 02
32	Gemeinkosten	Stampfbetonerzeugung	Anlagen-h	15.1		GSB	SST			23 01
33	Gemeinkosten	Nacharbeiten		m^2	15.2	GSB	SST			23 02
34	Gemeinkosten	Transporte		t	16.1	GSB	SST			23 01
35	Gemeinkosten	Paletten		Pal	16.2	GSB	SST			23 02
36	Gemeinkosten	Produktionsboden		%m^2	17.1	GSB	SST			23 01
37	Gemeinkosten	Dampfanlage		m^3	17.2	GSB	SST	3x		23 06
38	Summe Sonderkosten Fertigung						5)			
39	Betrieb allgemein (Zl 24+25+26)			h	18.1	GSB	SST	3x		00
40	Bruch, Verlust (ganzzahlig!)			%			SST	6x		23 06
41	Gemeinkosten Verw.u.Vertrieb			%	18.2	automatisch!				
42	Su Sonderkosten Verw.u.Vertrieb						5)	SST		16 22 00
43	Löhne Verladung			h	19.1	GSB	SST			23 01
44	Gemeinkosten Verladung			t	19.2	GSB			6)	
45	Gewinn und Risiko			%	20.1	automatisch!				
46	leere Magnetkarte beidseitig einschieben!					für neuen Beginn zurück zu Schritt 1!			130	max. Zl Nr.

1) ganze Zahlen ohne Dezimalpunkt, Dezimalzahlen ohne führende 0 vor Dezimalpunkt setzen!
2) GSB 9-Anweisung - 1 Programmzeile - nur bei 100-facher Wertangabe setzen!
3) SST-Tasten in allen Fällen - auch bei Leermeldungen - drücken!
4) Zeilennummer und Tastencode ergänzen!
5) bei Leermeldung I DEL drücken, für Leermeldungen keine 0 setzen!
6) wenn grösser als max. Zeilennummer, Datenkontrolle und neuer Beginn bei Schritt 1!

ta 1978 06 02

Beispiel C: **Mengenauflösung**

Im vorigen Beispiel hatten wir im Rahmen einer automatischen Kalkulation die Artikel-Stammdaten in ein Daten-Programm gestellt. Es liegt daher nahe, diese Daten-Programme auch für einen anderen Verwendungszweck heranzuziehen; die Mengenauflösung bietet sich an. Unter dem Begriff „Menge" sind natürlich alle Kostenarten zu verstehen, sodaß mit dem vorliegenden Operationsprogramm die Soll-Werte jeder einzelnen Kostenart, wie Material, Lohnstunden, Gemeinkostenschlüssel berechnet werden können.

Für den in Frage stehenden Zweck benötigen wir einen anderen Operationsteil. Nach Einlesen der Artikel-Datenprogramme sind nur noch die jeweilige Anzahl der erzeugten Artikel einzugeben, worauf die Gesamtsummen errechnet werden. Dabei ist es möglich, da die Speicher hier keine Verrechnungssätze zu tragen haben, diese zur Abspeicherung zu verwenden. Damit können sogleich die Summen der Soll-Werte für eine Artikelgruppe ermittelt werden. Wir müssen aber hier genauest die möglichen Größenordnungen der Ergebnisse vorher abschätzen, weil die Speicher wegen der großen Anzahl an Kostenarten, zur Abspeicherung der Kostenartensummen geteilt werden müssen und je Kostenart nur fünf wertführende Ziffern zur Verfügung stehen. Dabei ist auch der Frage der Stellung des Dezimalpunktes erhöhte Aufmerksamkeit zu widmen. Zur Ausgabe der Ergebnisse rufen wir die Summen der Kostenarten – Menge Material, Menge Lohnstunden, Menge Kostenstellenschlüssel – mit einer eigens hierfür definierten freien Funktionstaste ab.

Wir müssen dabei nun wie folgt vorgehen:

1. Einlesen des Operationsprogrammes Mengenauflösung.
2. Programm starten.
3. Taste drücken, die die Zuordnung zweier Programme nebeneinander im Programmspeicher ermöglicht.
4. Einlesen des Daten-Programmes.
5. Start der Verarbeitung.

Wiederholungen sind ab Punkt 3 vorgesehen. Es hängt von der Größenordnung der Ergebnisse ab, ob sofort nach einem Artikel, nach einer Artikelgruppe oder erst nach Durchlauf sämtlicher Artikel gelistet werden muß. Im übrigen erscheint bei der Listung zur besseren Leserlichkeit der Ergebnisse vor der jeweiligen Summe zuvor der zugehörige Kostenartenschlüssel, den wir schon bei der Kalkulation nach Beispiel A kennengelernt haben; er wird vor der Summe angedruckt. In der Kombination der Beispiele 1 bis 3 eröffnen sich für bestimmte Größenordnungen Möglichkeiten, die bislang nur größeren Rechnern offen standen. So ist es denkbar, daß derartige Lösungen, die man sich selbst erarbeiten kann, für Betriebe bestimmter Größenordnungen durchaus zu einer Dauereinrichtung werden können. Sollte dann doch eine größere DV-Lösung in eine wirtschaftlich greifbare Nähe rücken, hätte der Anwender den großen Vorteil, bereits Methoden zu besitzen, die als Vorlage für die größere Lösung dienen können.

Das für die Mengenauflösung verwendete Formblatt stimmt im Aufbau mit dem zur Aufnahme der Artikelstammdaten konzipierten Vordruck überein.

Um zu verhindern, daß sich in einem Speicher gemeinsam abgestellte Kostenartensummen gegenseitig beeinflussen, kann als Sicherung gegen die Überschreitung der maximal zulässigen Summengröße eine entsprechende Abfrage in das Programm eingebaut werden.

Dipl. Ing. Gerfried Tatzl, WIV Mörikestrasse 17, A 8052 Graz - Wetzelsdorf	MENGENAUFLÖSUNG	Datum

für

Schl.	Kostenart		Eht.	Menge	Untersumme
2.1	Mischerkosten	Zentral - Mischanlage	ME		
2.2		Vorsatzbeton - Mischanlage	ME		
3.1	Bindemittel	PZ 275	t		
3.2		PZ 375	t		
4.1		PZ 475	t		
4.2		Weisszement	°/₀ kg		
5.1	Zuschlagstoffe	Brechsand 0/2	t		
5.2		Splitt 2/5	t		
6.1		Splitt 5/8	t		
6.2		Sand 0/1	t		
7.1		Sand 1/4	t		
7.2		Kies 4/8	t		
8.1		Kies 8/16	t		
8.2		Marmorsplitt	t		
9.1	Bewehrung	Rippentorstahl RT 50	°/₀ kg		
9.2		Rippentorstahl RT 60	°/₀ kg		
10.1		Betonrundstahl St 37	°/₀ kg		
10.2		Baustahlgitter	°/₀ kg		
11.1	Fertigungslöhne	Eisenbiegen und Erzeugen	h		
11.2	(Mann - h)	Nacharbeiten	h		
12.1		Transporte	h		
12.2	Gemeinkosten	Eisenbiegerei	h		
13.1	(Anlagen - h)	Bodenfertiger	h		
13.2		Plattenanlage	h		
14.1		Ringmaschine	h		
14.2		Rütteltisch	h		
15.1		Stampfbetonerzeugung	h		
15.2		Nacharbeiten	m^2		
16.1		Transporte	t		
16.2		Paletten	Pal.		
17.1		Produktionsboden	°/₀ m^2		
17.2		Dampfanlage	m^3		
18.1		Betrieb allgemein (11.1+11.2+12.1)	h		
19.1	Löhne Verladung		h		
19.2	Gemeinkosten Verladung		t		

Beispiel D: **Kalkulatorische Kosten**

START		% G,B,M,Z	K	f_z, f_l
n, W	L, SuL, S	BERECHNUNG	DATÜB	AUS SUM

Unabhängig von der steuerlichen Behandlung der Anlagegüter hinsichtlich der Berechnung der Abschreibungen für Abnutzung gibt es bekanntlich auch die kalkulatorische Version für betriebsbuchhalterische Zwecke, wo die tatsächlichen Verhältnisse im Hinblick auf die Nutzungsdauer eines Anlagegutes berücksichtigt werden.

Wir haben wieder die Darstellung des Programmstreifens an die Spitze gestellt; in der nachfolgenden Erläuterung der einzelnen Tastenfunktionen wird die Problematik bereits weitestgehend angeschnitten, die sich um die Berechnung der kalkulatorischen Kosten rankt.

START: Eine Startfunktion ist erforderlich. Hier werden einige Summenspeicher gelöscht und damit zur Aufnahme neuer Daten bereit gestellt.

% G,B,M,Z: Mit dieser Bezeichnung soll angedeutet werden, daß wir die folgenden Prozentsätze einzugeben haben, die mit G, B, M und Z bezeichnet werden. Wir benötigen zur Ermittlung der Wiederbeschaffungswerte für Gelände (G), Bauten (B) und maschinelle Anlagen (M) die Aufzinsungsfaktoren. Dazu kommt noch der Zinssatz für die Ermittlung der Kapitalverzinsung (Z).

K: Es ist einzusehen, daß wir nicht für jede Anlagengruppe in gleicher Weise in der Berechnung der kalkulatorischen Kosten vorgehen können. Hier wird die Anlagenkennziffer K eingegeben, die diese Berechnung steuert. In diesem Programm sind deren vier vorgesehen und zwar

- 0 für Gelände;
- 1 für Bauten;
- 2 für Formeneinrichtungen zu maschinellen Anlagen und
- 3 für die maschinellen Anlagen selbst.

Mit der Anwahl der Anlagenkennziffer ist die Verarbeitungsrichtung festgelegt, die das Programm im aktuellen Fall zu gehen, besser „eilen" hat.

f_z, f_l: Jede Anlagenkennziffer berücksichtigt jene Faktoren automatisch, nach der zeit- und leistungsabhängige Abschreibungen berechnet werden. Hier ist die Möglichkeit gegeben, von dieser Norm abzuweichen, indem besondere Anteilsfaktoren f_z und f_l eingegeben werden können.

n, W: An dieser Stelle werden die tatsächliche Nutzungsdauer in Jahren und der letzte Wiederbeschaffungswert eingegeben.

L, SuL, S: Für maschinelle Anlagen und Formeneinrichtungen muß der jeweils letzte Stand der Nutzung eingegeben werden, da beispielsweise die leistungsbedingten Abschreibungen im Verhältnis von Soll-Leistung zu Ist-Leistung berechnet werden; dabei bedeuten: L die Leistung im abgelaufenen Rechnungsjahr; SuL die Summe der Leistung vom Beginn der Nutzung und S die Soll-Leistung für ein Rechnungsjahr bzw. für die gesamte Lebensdauer bei Formeneinrichtungen.

BERECHNUNG: Mit dieser Taste wird die Berechnung ausgelöst; ausgedruckt werden

1. der neue Wiederbeschaffungswert;
2. die neue Gesamtleistung SuL;
3. der Nutzungsgrad für das laufende Rechnungsjahr;
4. der Anteil der zeitabhängigen Abschreibung A_z;
5. der Anteil der leistungsabhängigen Abschreibung A_l;
6. der Gesamtwert der Abschreibungen A_g;
7. der Variator der Abschreibungen; der leistungsabhängige Anteil der Abschreibungen wird langfristig gesehen als variable Kostenpost angesehen;
8. die Zinsen.

Sollte irgendetwas schief gelaufen sein, kann die Berechnung vor einer Datenübernahme wiederholt werden.

DATÜB: Erst wenn man sich der Sache sicher ist, sollte diese Taste gedrückt werden, denn mit ihr werden der neue Wiederbeschaffungswert, zeit- und leistungsabhängige Abschreibung und Zinsen in die jeweiligen Summenspeicher transferiert.

AUS SUM: Diese zuvor genannten Gesamtsummen werden mit dieser Taste abgerufen und angedruckt, wobei zusätzlich noch die Gesamtsumme der Abschreibungen berechnet und angedruckt wird. Dazu eine Bemerkung: Es wäre unsinnig, auch die jeweilige Gesamtsumme der beiden Anteile der Abschreibungen abzuspeichern. Sie kann jederzeit aus der Summe der beiden Einzelbeträge berechnet werden!

Nun zur Darstellung der verwendeten Formeln und Bezeichnungen.

Legende

W	=	alter Wiederbeschaffungswert;
W'	=	neuer Wiederbeschaffungswert;
L	=	Ist-Leistung im Rechnungsjahr;
SuL	=	Summe der Ist-Leistung bis vor Rechnungsjahr;
SuL'	=	Summe der Ist-Leistung einschl. Rechnungsjahr;
S	=	Soll-Leistung pro Rechnungsjahr; bei Formen für gesamte Lebensdauer;
η	=	Auslastungsfaktor;
p_G	=	Aufzinsung Grundstücke %;
p_B	=	Aufzinsung Gebäude %; (Baukosten-Index);
p_M	=	Aufzinsung Formen und maschinelle Anlagen % (Geräte-Index);
p_Z	=	Kapitalverzinsung %;
n	=	Nutzungsdauer in Rechnungsjahren;
f	=	Aufzinsungsfaktor für Wiederbeschaffung;
f_z	=	Anteilsfaktor für zeitbedingte Abschreibung;
f_l	=	Anteilsfaktor für leistungsbedingte Abschreibung;
A_g	=	gesamte Abschreibung;
A_z	=	zeitbedingte Abschreibung;
A_l	=	leistungsbedingte Abschreibung;
K	=	Anlagen-Kennziffer;
V	=	Variator Abschreibung;
Z	=	Zinsen

Formeln

$W = W \times f;$

$f = 1 + \frac{p}{100}$; für p gilt p_G, p_B und p_M

$\eta = \frac{L}{S};$

$SuL' = SuL + L;$

$A_z = f_z \times \frac{W'}{n}$; (= 0 bei K = 0,2)

$A_l = f_l \times \frac{W'}{n} \times \eta$; (= 0 bei K = 0,1)

$A_g = A_z + A_l;$

$V = \frac{A_l}{A_z} \times 100;$

$Z = \frac{W}{2} \times \frac{p_Z}{100}$; gilt nicht für Grundstücke!

$Z = W \times \frac{p_Z}{100}$; gilt nur für Grundstücke!

K = 0 Gelände
1 Bauten
2 Formen
3 masch. Anlagen

Zu den letzten beiden Formeln für die Berechnung der Zinsen ist anzumerken, daß es für Grundstücke keine Abschreibungen gibt und daher die Verzinsung vom vollen Betrag anzusetzen ist, d. h. vom vollen Wiederbeschaffungswert. Die Reihenfolge der beschriebenen freien Funktionstasten entspricht auch der Reihenfolge der aufzuführenden Vorgänge.

Bei dieser Art von Anlagenbuchhaltung sind in die von Hand zu führenden Anlageblätter die jeweiligen Daten nachzutragen; die Blätter dienen somit als Beleg. Hier liegt die Zeiteinsparung nur bei der Ausführung der Rechenvorgänge selbst.

Aus der Diskussion der Details ist die Funktion dieses Programms wohl eindeutig hervorgegangen. Unsere Erörterungen konnten sich aber nur auf die Durchführung des Rechenansatzes und des Ablaufes bemühen, nicht auf rein sachbezogene Fragen der Anlagenbuchhaltung selbst.

Beispiel E: **Kalkulatorische Kosten automatisch**

Auch hier lag die Versuchung nahe, die Magnetkarten zu Konto-Karten umzufunktionieren und die Ermittlung der kalkulatorischen Kosten nach dem vergangenen Beispiel D auch automatisch ablaufen zu lassen. Zur Bewältigung dieser Problemstellung benötigen wir einmal das Programm selbst, das hier natürlich anders aufgebaut sein muß. Mit diesem Programm geben wir auch die vier Aufzinsungs-Prozentsätze ein. Je Anlage brauchen wir natürlich eine Magnetkarte oder ein „Magnetkonto". Auf diesen sind die zuvor beschriebenen Daten gespeichert. Zusätzlich sind natürlich auch die jeweils charakteristischen Anlagen-Daten, wie Anlagen-Kennziffer, Anlagen-Nummer, zu belastende Kostenstelle, sowie die Anteilsfaktoren der zeit- und leistungsabhängigen Abschreibungen gespeichert. Diese Daten geben wir wieder mit einem Ladeprogramm oder von Hand ein.

Die Aufzinsungsfaktoren sind dabei ganzzahlig und zweistellig einzugeben, da sie gemeinsam in einem Speicher untergebracht werden müssen.
Wir können den Vorgang wie folgt beschreiben:

1. Einlesen der Programmkarte.
2. Eingabe der vier ganzzahligen Aufzinsungsprozentsätze.
3. Einlesen der Anlagenkarte.
4. Beginn der Verrechnung. Sollte der Rechner bei Anzeige der Zahl 0 anhalten, wissen wir, daß wir an dieser Stelle bei Formen und maschinellen Anlagen die Ist-Leistung eingeben müssen.

Der Andruck der Ergebnisse je Anlage erfolgt in ähnlicher Weise, wie zuvor beschrieben, aber mit kleinen Änderungen; wir erhalten angedruckt

1. die Inventar-Nummer;
2. die Nummer jener Kostenstelle, der die Anlage zugerechnet wird;
3. den neuen Wiederbeschaffungswert,
4. die Auslastung der Anlage;
5. die zeitabhängige Abschreibung;
6. die leistungsabhängige Abschreibung;
7. die Gesamtabschreibung;
8. den Variator der Abschreibung und
9. die Zinsen.

Wegen Speicherplatzmangel werden hier nur drei Summenwerte für die Wiederbeschaffungswerte, die gesamten Abschreibungen und die Zinsen geführt und am Ende der Verarbeitung ausgedruckt.

Da die Aufzinsungsprozentsätze sich ja Jahr für Jahr ändern, können sie nicht fix auf den „Anlagekonten" abgespeichert werden. Nun würden diese %-Sätze beim Einlesen der zweiten Anlagenkarte verloren gehen, wenn nicht bei Benützung der umgekehrten polnischen Notation als Rechenlogik die vier Stack-Speicher zur Rettung oder Datensicherung herangezogen werden könnten. Auch ein wertvolles Detail am Rande! Am Ende bei Datenübernahme der neuen Salden auf die Magnetstreifen, werden die vier %-Sätze und die drei Gesamtsummen für Wiederbeschaffungswerte, Abschreibungen und Zinsen in die vier Stackspeicher X, Y, Z und T übernommen und bei Beginn einer neuen Verarbeitung nach Einlesen des nächsten „Kontos" in den alten Speichern abgestellt.

Die Beendigung einer Verarbeitung wird in der Anzeige mit „Crd" dargestellt; es wird die alte Datenkarte verlangt und der neue Kontostand wird übernommen. Nach Einschieben einer Seite des Anlagenkontos erscheint „Crd" ein zweites Mal. Damit wird angedeutet, daß auch die zweite Seite des Magnetstreifens einzuschieben ist. Dabei bleiben wohl die vier vorerwähnten Speicher – Prozentsätze, Summen, Wiederbeschaffungswerte, Abschreibungen und Zinsen – mit den jeweils letzten Ständen auf dem Magnetstreifen.

Dies ist ohne Bedeutung, da die Inhalte dieser Speicher bei der nächsten Verarbeitung ohnehin von anderen Werten überschrieben werden.

Bei einer größeren Anzahl von Anlagenkarten kann das Programm selten in einem Zug sämtliche anstehenden Anlagenkarten verarbeiten. Daher ist im Programm auch eine **Unterbrechung** vorgesehen. Die Taste C wird gedrückt und der letzte Stand der Dinge wird auf eine leere Magnetkarte „geschrieben". Auch hier ist der Hinweis für eine Datenübernahme zur Zwischenspeicherung mit der Anzeige „Crd" gegeben. Der neue Beginn am nächsten Tag oder zu einem anderen Zeitpunkt nach der Unterbrechung beschränkt sich auf das neuerliche Einlesen des Programmes, der Datenkarte mit den Zwischenergebnissen, worauf nach Betätigung der Start-Taste die Verarbeitung mit dem Einlesen der nächsten Anlagenkarte fortgesetzt werden kann. Erst wenn die letzte Anlage verarbeitet worden ist, wird der Summenabruf ausgelöst.

Die ausgedruckten Werte sind dann von Hand weiter zu verarbeiten zu Kostenstellen-Summen bzw. Kostenträger-Summen, soweit es sich hier um Formeneinrichtungen handelt, die ja direkt auf Kostenträger verrechnet werden.

Als Ergänzung mag dienen, daß nach Andruck der Inventar-Nummer die Zahl 0 kommt, wenn die Anlage vollständig abgeschrieben ist.

Beispiel F: **Auftragsbearbeitung**

Auftragsbearbeitung

START m/oR	TYP	TYP	TYP	TYP
RABATT		SONDERART	STANDART	ABSCHLUSS

Ladeprogramm

START	MWST	MAX.DAT		
INDEX	AUS AB IND	AUS FÜR IND	EIN P	EIN G, P

Beim Problem einer Auftragsbearbeitung geht es darum, mit standardmäßig gespeicherten Wertepaaren für Gewicht und Preis (Listenpreis) von Artikeln durch Eingabe des Artikel-Index und der Menge je Position den Gesamtpreis zu erhalten. Nach Eingabe sämtlicher Positionen, wird absummiert, d. h. der Nettobetrag ausgedruckt, die Mehrwertsteuer berechnet sowie der Bruttobetrag ausgedruckt. Wichtig war auch, das gesamte Liefergewicht dabei festzuhalten. Desgleichen war gefordert, daß ein Rabatt entweder zeilenweise berechnet oder erst im gesamten vor Aufrechnen der Mehrwertsteuer abgezogen werden sollte. Hierfür könnte man Datenkarten entwickeln mit jenen Produktkombinationen, die meist auch zusammen verkauft werden. Außerdem erschien es günstig, je Produktkombination ein eigenes Auftragsblatt zu entwerfen, das genau dieser Produktkombination entsprach. Auf diesem Blatt waren die maximal 20 möglichen Produkte mit ihrem Index angeführt.

Für dieses Beispiel haben wir auch einmal das **Laden von Daten** mit in unsere Betrachtungen einbezogen. Wir wollen auch gleich damit beginnen.

START: Nach Einlesen dieses Programmes ist es auch zu starten, um sämtliche Speicher frei zu machen; der erste Index wird eingestellt.

MWST: An dieser Stelle wird der Mehrwertsteuersatz eingegeben.

MAX.DAT: Vom Rechner her ist die Speicherkapazität unter Berücksichtigung der übrigen Bedürfnisse auf 20 Artikel beschränkt. Es können aber auch weniger Artikel sein; denn nach Anwahl des letztmöglichen Index wird die Abspeicherung der eingegebenen Daten auf der Datenkarte automatisch unter Anzeige von „Crd" verlangt und bei der Verrechnung werden alle Zahlen, die außerhalb der so bestimmten Indices liegen, vom Rechner abgewiesen.

INDEX: Für Änderungen ist eine gesonderte Anwahl eines Index erforderlich.

AUS AB IND: Zur Kontrolle der Eingaben können ab einem angewählten Index sämtliche Wertepaare ausgedruckt werden.

AUS FÜR IND: Das gleiche gilt in dem Fall nur für den einen angewählten Index.

EIN P: Über diese Taste kann ein Preis allein eingegeben werden, dies ist im Falle von Preisänderungen erforderlich, da bei dieser Gelegenheit wohl kaum die Gewichte der Artikel zu ändern sind.

EIN G, P: Über diese Taste werden die Wertepaare Gewicht und Preis eingegeben; der Index wird dabei automatisch weitergestellt.

Die Auftragsbearbeitung hat folgendes Aussehen:

START m/oR: Das Programm kann sowohl mit, als auch ohne Rabatt gestartet werden; im zweiten Fall wird mit dem gespeicherten Rabatt weitergearbeitet.

TYP: Über diese Tasten ist es möglich, für bestimmte Artikel die Mengen durch Eingabe vom Rechner ermitteln zu lassen. In diesen Fällen wird durch Eingabe bestimmter Parameter die Menge berechnet. Zum Beispiel kann aus einer Gesamtfläche, begrenzt durch Länge und Breite, und der Fläche einer Produkteinheit auf die gesamte Produktmenge geschlossen werden.

RABATT: Durch Betätigung dieser Taste kann jeweils von Gesamtrabatt auf Zeilenrabatt oder umgekehrt umgeschaltet werden.

SONDERART: Die Möglichkeit, auch nicht standardmäßig enthaltene Artikel verrechnen zu können, muß offen bleiben. Hier müssen wir anstelle des Index Gewicht und Preis eintasten.

STANDART: Über diese Taste sind Index und Menge, soweit sie nicht durch TYP berechnet wurden einzugeben. Außerdem ist an dieser Stelle auch der Prozentsatz eines Zeilenrabattes einzugeben, falls dieser eingangs vorgesehen wurde.

ABSCHLUSS: Mit dieser Taste wird die Auftragsberechnung im geforderten Sinne beendet.

Beispiel G: **Energieverbrauchskontrollen**

Im vorliegenden Fall wollen wir uns auf eine Darstellung eines Details aus der Stromabrechnung begnügen.

Nicht erst heute unter dem Eindruck, immer mehr Energie sparen zu müssen, sondern schon viel früher hätte das Bewußtsein, daß im allgemeinen Energie verschwendet wird, geweckt werden müssen. Notwendig wurde eine monatliche Eigenkontrolle schon aus dem Umstand heraus, daß die meisten Energieversorgungsunternehmungen aus Kostengründen auf eine jährliche Abrechnung übergegangen sind und im vorhinein gleiche Monatsbeträge vorschreiben.

Die Kontrolle setzt nun bei der monatlichen Selbstablesung ein und wird mit einer Gegenüberstellung des Energieverbrauches und beispielsweise der Produktionsmenge abgeschlossen.

Auch hier kann PROTAR wieder eine große Hilfe sein. Zur Erstellung der Programme sind zuvor die Energielieferungsverträge genau zu studieren und der augenblickliche Stand der Tarife festzuhalten.

Es wäre hier tatsächlich illusorisch, die Technik der Abrechnung darzustellen, einerseits, da der Rechenvorgang zur Ermittlung der Bezugskosten im Prinzip überall gleich ist: Zählergebühr + Leistungs- oder Grundpreis und Arbeitspreis nach verbrauchten Mengen.

Andererseits sind die Voraussetzungen, wie beispielsweise ein Grund- oder Leistungspreis berechnet wird, regional stark unterschiedlich.

Wir wollen aber, wie angekündigt, ein Detail aus der Stromabrechnung darstellen, wie er von den Grazer Stadtwerken zur Berechnung des Leistungspreises verwendet wird.

Ein Betrieb ist für eine maximale Leistung von 240 KW ausgelegt. Die Stadtwerke verlangen eine Mindestverrechnungsleistung in der Höhe von 60% das sind 144 KW monatlich. Ab dieser Leistung wird mit einem Maximumzähler der Mehrbezug an Leistung festgehalten. Nun gibt es die Regelung, daß für die tatsächliche Verrechnungsleistung ein Durchschnittswert aus den höchsten drei Monatshöchstlasten eines Jahres herangezogen wird. Bleiben jedoch diese Höchstlasten unter 144 KW, so gilt die Mindestverrechnungsleistung als Grundlage, liegen ein oder zwei Höchstlasten unter dieser Grenze, so werden zur Mittelwertbildung nicht die tatsächlichen unter dieser Grenze liegenden Werte, sondern diese Grenzen selbst herangezogen, auch dann, wenn der Mittelwert aus den tatsächlichen drei Höchstlasten über dem Mindestverrechnungswert liegen würde.

Ein Beispiel: Die drei Höchstlasten betragen 134, 138 und 166. Durch diesen Ausreißer von 166 KW kommt das tatsächliche Mittel bei 146 KW zu liegen. Nun sagt aber der Tarifvertrag, daß in diesem Fall das Mittel aus 144, 144 und 166 zu berechnen ist, Ergebnis 151 gerundet. Die Grundlage der Berechnung des Leistungspreises liegt nun tatsächlich bei 151 KW anstelle 146 KW. Nun wird dieser Mittelwert jeden Monat – außer in den Monaten Januar und Februar – berechnet. Damit kommen wir auf das Grundproblem unserer Betrachtungen in diesem Beispiel. Es ist jedesmal abzuprüfen, ob der neue Wert, d. h. das neue Monatsmaximum an Leistungsbezug – nicht zu verwechseln mit der bezogenen Arbeit in kwh! – größer ist als einer jener drei Werte, die zuletzt den Durchschnittswert gebildet haben. Falls das so ist, muß der neue Wert in diese Reihe neu eingeordnet werden.

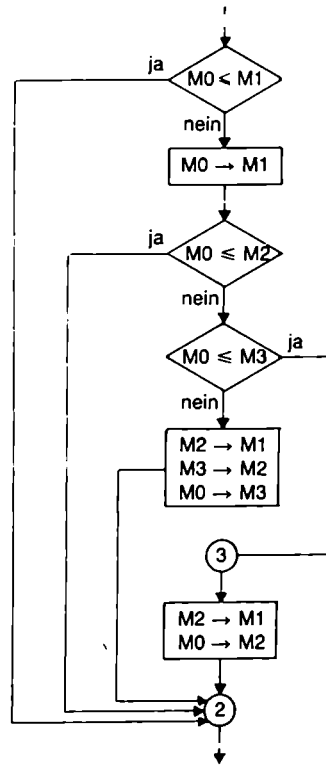
Die Sache sieht relativ kompliziert aus, aber im Grundproblem selbst sehen wir schon klar. Nun ergibt sich der Fall, daß bei höherem neuen Durchschnittswert des Leistungsbezuges für die vor dem Abrechnungsmonat liegenden Monate eine Nachverrechnung erfolgt und zwar werden jene KW nachberechnet, die sich aus der Differenz neuer Durchschnittswert abzüglich alter Durchschnittswert multipliziert mit der Anzahl der vor dem Abrechnungsmonat liegenden Monate ergeben.

Der eingegebene Zählerstand des Maximumzählers wird erst einmal mit der Zählerkonstanten multipliziert und dabei auf volle KW kaufmännisch gerundet. Und dann geht diese Zahl als erstes in die Abfrage der drei bisherigen Höchstwerte. Dieses Detail wollen wir uns nun im Block-Diagramm und in der für UPN geschriebenen Befehlsfolge ansehen.

Teil-Blockdiagramm

Einordnen des neuen Maximums in die drei bestehenden Höchstmaxima;
M0 = neues Maximum;

M1 kleiner M2 kleiner M3 sind die bestehenden drei Höchstwerte.



weiter im
Programm

Speicherbelegung:
M0 = Speicher 4
M1 = Speicher 5
M2 = Speicher 6
M3 = Speicher 7

Und nun die Tastenfolge:

LBL 1	Das Programm kommt zur Einordnung des neuen Wertes M0.
RCL 5	kleinster alter Höchstwert
RCL 4	neuer Wert in der Anzeige
x ≤ y	Der neue Wert wird mit dem kleinsten der bisherigen drei Höchstwerte verglichen.
GTO 2	Ist er nicht größer, wird das Programm fortgesetzt;
STO 5	andernfalls wird M0 vorerst einmal im Speicher für M1 abgespeichert.
RCL 6	mittlerer alter Höchstwert
RCL 4	neuer Wert
x ≤ y	Nun wird der neue Wert M0 dem zweithöchsten der bisherigen drei Höchstwerte gegenübergestellt.
GTO 2	Ist er nicht größer als dieser, wird das Programm beim Label 2 fortgesetzt.
RCL 7	größter alter Höchstwert
RCL 4	neuer Wert
x ≤ y	Als letztes erfolgt der Vergleich des neuen Wertes M0 mit dem größten der bisherigen drei Höchstwerte.
GTO 3	Ist er nicht größer als dieser, wird der bisherige zweithöchste Wert zum kleinsten, während der neue Wert an die Stelle des neuen zweithöchsten tritt.
RCL 6	Im Falle, daß der neue Wert M0 größer als der alte größte der drei Höchstwerte ist, rutschen alle alten Werte je eine Stufe tiefer, während M0 nun an die Stelle des neuen größten Höchstwertes tritt. Der alte kleinste der drei Höchstwerte verschwindet somit.
STO 5	
RCL 7	
STO 6	
RCL 4	
STO 7	
GTO 2	Nach dieser Umspeicherung wird das Programm beim Label 2 fortgesetzt.
LBL 3	
RCL 6	
STO 5	
RCL 4	
STO 6	

LBL 2 Ab hier erfolgt die Fortsetzung im Programm nach der Einordnung des neuen Monatsmaximums.

Die Fortsetzung im Programm besteht nun darin, daß der neue Mittelwert mit dem Mindestverrechnungswert verglichen wird, worauf die Entscheidung über den weiteren Ablauf im Rechner gefällt wird.

Ähnliche Tarifprobleme gibt es auch in der Gasverrechnung. Hier hat man eine Preisstaffelung vorgenommen derart, daß für den Bezug der ersten 350 m^3 im Jahr der höchste Tarif, für den Bezug zwischen 351 und 900 m^3 ein mittlerer und ab 901 m^3 der niedrigste Tarif der Abrechnung zugrunde gelegt wird. Das Programm muß also hier eine Stelle passieren, in der der aktuelle Gesamtverbrauch kontrolliert wird. Es ist durchaus möglich, daß zu Jahresbeginn ein Verbrauch nach zwei Tarifen abzurechnen ist; diesem Umstand muß dann auch unser Programm Rechnung tragen.

Allgemein ist zur Verrechnung der Energiekosten zu sagen, daß Tarife, Zählerstände und Verbrauch auf einer Datenkarte aufgezeichnet und die Summen gleichzeitig fortgeschrieben werden. Dabei bedient man sich des Vater-Sohn-Prinzips, d. h. die letzte Verrechnung ist immer wiederholbar. Würde uns ein Fehler in der aktuellen Abrechnung passieren, so wäre bei einer einzigen Datenkarte der letzte Zustand kaum oder nur sehr schwer zu rekonstruieren. Deshalb werden die neuen Werte immer auf den zweiten Datenstreifen aufgezeichnet, von wo dann die Angaben für die nächstfolgende Verrechnung heruntergeholt werden.

Beispiel H: **Auswertung von Zeitaufnahmen**

START	LG?			STORNO
ENDE	DSP EING	→ N	← H	→ H, SuA

Wir beginnen wieder mit der Darstellung der einzelnen Programmfunktionen.

START: Es werden Speicher gelöscht, der Index auf die Zahl 4 gestellt – die Speicher 0 bis 3 werden später anderweitig verwendet – und 20 Speicher zur Aufnahme von Zeilensummen bereitgestellt.

LG ?: Hier wird entschieden, ob nach Beendigung der Eingabe des letzten Zeitzyklus die Ausgabe der Mittelwerte der einzelnen Zeiten mit der Eingabe eines Korrekturfaktors, des Leistungsgrades, während eines Programmstops vorgenommen werden soll.

STORNO: Die zuletzt eingegebene Fortschrittszeit kann storniert werden, gleichgültig über welche Taste die Eingabe erfolgt ist. Hier bewährt sich die Flagtechnik, da aufgrund von Flags nach der Eingabe bei der Stornierung der Weg der Eingabe vom Rechner festgestellt werden kann. Selbst war man nicht sicher, über welche Taste die Eingabe der letzten Fortschrittszeit erfolgt ist, der Rechner weiß dies. So kann man seine eigenen Unsicherheiten beseitigen.

ENDE: Abruf der Durchschnittswerte der Zeilen, d. h. der mittleren Zeiten jedes Arbeitsganges und der Gesamtzeiten nach Haupt- und Nebenzeiten getrennt.

DSP EING: Mitunter kann es aus welchem Grund auch immer erforderlich werden, die zuletzt eingegebene Fortschrittszeit über die Anzeige abzurufen.

→ N: Eingabe der Fortschrittszeit im Falle des Vorliegens einer Nebenzeit.

← H: Nach Eingabe einer Hauptfortschrittszeit springt der Zeilenindex jeweils um 1 weiter; mit dieser Taste kann aber in die letzte Zeile hineinaddiert werden. Dies ist erforderlich, wenn inmitten eines Arbeitsganges eine Unterbrechung notiert wurde.

→ H, SuA: Hier erfolgt die Eingabe der Hauptfortschrittszeiten. Dabei wird zur Absummierung der Zeiten eines gesamten Zyklus (Spaltensummierung) jene Eigenschaft eines Rechners benutzt, die es gestattet zu entscheiden, ob eine Zahleneingabe erfolgt ist oder ob diese Taste ohne vorangegangene Zahleneingabe bestätigt wurde; die Spaltensumme wird im zweiten Fall angedruckt. Damit wird die Auswertung einer Zeitaufnahme wesentlich erleichtert, da jeder Zeitwert nur einmal in den Rechner einzutasten ist, wobei zusätzlich Haupt- und Nebenzeiten getrennt erfaßt, Spaltensummen gebildet und gleichartige Arbeitsgänge in Zeilensummen erfaßt werden. Die Durchschnittszeiten werden dann durch die Division der Zeilensummen geteilt durch die Anzahl der gebildeten Spaltensummen (Anzahl der gemessenen Arbeitszyklen) ermittelt. Zum Abruf der Spaltensumme kann aber auch eine noch freie Funktionstaste herangezogen werden.

Im einzelnen wird wie folgt dargestellt vorgegangen:

1. Nach Eingabe der Fortschrittszeit wird der Zeitunterschied zur letzten Eingabe berechnet und damit die Zeitdauer des gemessenen Arbeitsganges bestimmt. Je nach gewählter Eingabetaste erfolgt die Abspeicherung als Haupt- oder Nebenzeit.
2. Nach Aufrechnung sämtlicher Arbeitsgänge eines Zyklus wird dieser absummiert.
3. Es werden Haupt- und Nebenzeitsummen gebildet.

4. Bei Berechnung der Zeitunterschiede werden diese in den entsprechenden Arbeitsgangspeichern zu Zeilensummen aufsummiert, die dann zur Mittelwertbildung herangezogen werden.
5. Es besteht die Möglichkeit, die jeweils letzte Eingabe zu stornieren, wobei es ohne Belang ist, über welche der drei Eingabetasten die Eingabe tatsächlich erfolgt ist.
6. Wurde die Arbeit während eines Arbeitsganges unterbrochen, kann über eine besondere Taste in die gleiche Zeile hineinaddiert werden.
7. Spätestens nach Absummierung des letzten Zeitzyklus kann noch entschieden werden, ob die Durchschnittszeiten mit einem Leistungsgrad zu multiplizieren sind.
8. Abruf der Ergebnisse.

Dieses Programm ist eine große Hilfe für jene, die Zeitaufnahmen auszuwerten haben, wobei als Einschränkung zu wiederholen ist, daß ein Arbeitszyklus bis zu 20 Arbeitsgänge haben kann.

Beispiel I: **Tabellenaufrechnung**

START	SPALTEN			
INDEX	AUS SUMZ	STORNO		WERT

Dieses Beispiel wird bewußt an einem Rechner der Fa. HEWLETT-PACKARD, dem Modell HP 97 gezeigt, um ein komplettes Programm mit sämtlichen Unterlagen präsentieren zu können. Im Falle unseres Beispiels sind das

- Programmbeschreibung,
- Programmlistung und
- Bedienungsanleitung.

Zuvor mögen noch die einzelnen Funktionstasten etwas näher erläutert werden, etwas, was aus den Unterlagen vielleicht doch nicht ganz so deutlich hervorgeht.

START: Es werden die Register gelöscht, eine eventuell vorhanden gewesene Spaltenzahl für eine neue Tabellenaufrechnung gerettet und der Index auf die Ziffer 1 gestellt, d. h. auf die erste Spalte. Insgesamt können Tabellen bis zu 22 Spalten aufgerechnet werden, Beschränkungen nach Zeilen gibt es keine.

SPALTEN: Als Hilfe für eine automatische Ausgabe einer Zeilensumme wird die tatsächlich vorhandene Anzahl der Spalten, die die Zahl 22 nicht überschreiten darf, eingegeben.

INDEX: Eine Index-Anwahl ist erforderlich, wenn einige Spalten zu überspringen sind. Dies kann aber auch durch eine mehrmalige Eingabe des Wertes 0 bei WERT erfolgen.

AUS SUMZ: Die Zeile mit den Spaltensummen und der Gesamtsumme wird angedruckt.

STORNO: Auch hier ist eine STORNO-Taste von großem Vorteil, da bei der Eingabe vieler Zahlen Irrtümer und Fehleingaben nie ganz ausgeschlossen werden können. Ist eine Zeile dabei schon absummiert, wird im Falle einer Stornierung des letzten Wertes einer Zeile die Zeilensumme negativ angedruckt, worauf die Neueingabe erfolgen kann.

WERT: Hier erfolgt die Eingabe der einzelnen Werte einer Tabelle. Ist der letzte Wert einer Zeile nicht in der letztmöglichen Spalte gelegen, ist zur Absummierung der Zeile die Eingabetaste ohne Zahlenangabe zu betätigen. Haben wir in der letztmöglichen Spalte dagegen einen Wert einzugeben, erfolgt die Ausgabe bzw. der Andruck der Zeilensumme automatisch, worauf der Index wieder auf die erste Spalte eingestellt wird.

Wie im Falle der Auswertung von Zeitaufnahmen vermeiden wir es auch hier, eine Zahl zweimal „in die Hand" zu nehmen. Denn das wäre ja bei einer konventionellen Verarbeitung nicht zu vermeiden. Bilden Sie nach den Zeilensummen die Spaltensummen, müssen sie jeden Wert ein zweites Mal eintasten, was die Fehlermöglichkeit verdoppelt. Das zeitaufwendigste wäre nicht die zweite Eingabe ein und derselben Zahl, sondern das Suchen des Fehlers, wenn die Summe der Zeilensummen und die Summe der Spaltensummen, das ist die Zahl – in unserem Beispiel 110 – ganz rechts unten, nicht miteinander übereinstimmen.

Ein einfaches Beispiel, das immer wieder vorkommt.

Sie werden auch bemerkt haben, daß in den letzten Beispielen immer wieder die Möglichkeit der Stornierung der jeweils letzten Eingabe vorgesehen wurde. Speziell im Falle der Tabellenaufrechnung wäre es äußerst unangenehm, bei der vorletzten Zahl einen

Fehler zu begehen ohne die Möglichkeit einer Korrektur zu besitzen. Daher wählen wir eine automatische Rückführung der Rechnung auf den Stand vor der letzten Eingabe und programmieren ein Korrekturroutine. Diese werden in Standardprogrammen leider selten eingebaut. Eine automatische Korrekturmöglichkeit wird dann nicht benötigt, wenn wir uns über die Richtigkeit der Eingabe durch Anzeige oder Ausdruck des eingegebenen Wertes orientieren können und außerdem nicht an einen durch Index angewählten Eingabespeicher gebunden sind. In diesem Fall können wir den Irrtum durch die negative Eingabe des letzten Wertes richtig stellen. Hier würde uns der inzwischen um eine Position weitergerückte Index auch bei Anzeige des irrtümlich eingetasteten Wertes hindern. Wir müssen also auch den Index zurückstellen. Sicherlich können beide Vorgänge auch unabhängig voneinander abgewickelt werden. Bequemer und eleganter geht es jedoch mit Hilfe einer automatisch ablaufenden Stornierung, die über einen einzigen Tastendruck, den einer freien Funktionstaste, ausgelöst werden kann.

In der Anzeige erscheint nach jeder Eingabe nicht der eingegebene Wert, sondern der Index für die nächste anzusprechende Spalte. Sicherlich hätte man sich auch die Eingabe anzeigen lassen können. Dies stellt aber keine ausreichende Sicherheit dar, um nach Unterbrechungen den aktuellen Stand der Aufrechnung erkennen zu können.

So werden Sie kaum in der Lage sein, die Fortsetzung einwandfrei zu finden, wenn in der Zeile, in der Sie in der Eingabe unterbrochen wurden, zwei gleiche Werte haben und gerade dieser in der Anzeige aufscheint. Daher erschien die Angabe der Spalte sinnvoller; die jeweilige Zeile zu finden wäre kein Kunststück, da ja das Ergebnis einer Zeilensummierung immer angedruckt wird und Sie diese Zahl sicherlich auch bereits in die Tabelle eingetragen haben. Hätten Sie einen Rechner mit der Möglichkeit der Parallelität von mehr als einem Index zur Verfügung, könnten Sie mit zwei Indices arbeiten, indem Sie einen Index zur Kennzeichnung der Zeilen, den anderen zur Kennzeichnung der Spalten einsetzen. In der Anzeige lassen Sie dann beide Indices kombiniert durch den Dezimalpunkt getrennt erscheinen; zum Beispiel 5.7 würde bedeuten, daß als nächste Eingabe das 7. Element der 5. Zeile einzutasten wäre.

Ein triviales Problem, daß die Problemlöser in der DV nicht für wesentlich halten dürften, aber eines, über das sich im Zusammenhang mit der Nutzung von PROTARen nachgedacht zu haben sicherlich gelohnt hat.

Dipl. Ing Gerfried Tatzl, WIV
Mörikestraße 17,
A-8052 Graz-Wetzelsdorf

Programmbeschreibung

Datum 1979-03-08

HP 97

Programm - Titel TABELLENAUFRECHNUNG

Nr.

Programm - Verfasser wie oben

Programm - Beschreibung, Formeln, Bezeichnungen

Das vorliegende Programm erleichtert die Aufsummierung von Tabellen. Die für einen aktuellen Fall benötigte Anzahl von Spalten – im folgenden Beispiel 5 – kann angewählt werden, darf aber die Zahl 22 nicht überschreiten. Dadurch wird erreicht, daß nach Eingabe des jeweils letzten Wertes einer Zeile automatisch der Ausdruck der Zeilensumme erfolgen kann. Ist aber der letzte Wert einer Zeile vor Erreichung der letztmöglichen Spalte gegeben – im folgenden Beispiel ist dieser Fall in Zeile 3 gegeben – kann der Ausdruck der Zeilensumme vorzeitig durch Drücken der Taste E ausgelöst werden. Leerfelder können durch die Anwahl der als nächstes benötigten Spalte übersprungen werden; bei nur einem Leerfeld kann das Überspringen einer Spalte durch die Eingabe der Zahl 0 über die Taste E bewirkt werden. Die jeweils letzte Eingabe kann bei Vorliegen eines Eingabefehlers unmittelbar nach der irrtümlichen Einspeicherung storniert werden. Nach einer Eingabe wird grundsätzlich der Index jener Spalte angezeigt, für die die nächste Eingabe eines Tabellenwertes vorgesehen ist. Der Ausdruck der Spaltensummen muß von Hand ausgelöst werden. Im Anschluß daran kann sofort eine neue Tabelle aufgerechnet werden; allenfalls ist eine neue Dezimalstellenausgabe anzuwählen. Das Ausgabeformat muß nach dem Einlesen des Programmes nur dann über das Tastenfeld eingestellt werden, wenn eine von 0 Dezimalstellen abweichende Zahlendarstellung beim Ausdruck für eine Tabelle erforderlich ist.

Achtung: Die Storno-Taste darf niemals zweimal hintereinander betätigt werden! Bei Richtigstellung der letztmöglichen Zahl in einer Zeile wird die inzwischen ausgedruckte Zeilensumme ein weiteres Mal, aber negativ ausgedruckt. Nach Eingabe des richtigen Wertes wird die Zeilensumme nun richtig ausgedruckt.

Beispiel

	Spalte (1)	(2)	(3)	(4)	(5)	Summe
Zeile 1:	2	5	8	3	6	24
Zeile 2:		3		7	8	18
Zeile 3:	5	4		9		18
Zeile 4:	1	7	3	2	2	15
Zeile 5:					6	6
Zeile 6:	2	6	9	9	3	29
Summe	10	25	20	30	25	110

Folgende Tasten sind nach Einlesen des Programmes zu drücken:

(f) (A) 5 (f) (B);
2 (E) 5 (E) β (E) 3 (E) 6 (E) – Ausdruck 24;
0 (E) 3 (E) 0 (E) 7 (E) 8 (E) – Ausdruck 18;
5 (E) 4 (E) 0 (E) 9 (E) (E) – Ausdruck 18;
1 (E) 7 (E) 3 (E) 2 (E) 2 (E) – Ausdruck 15; 5 (A) 6 (E) – Ausdruck 6;
2 (E) 6 (E) 9 (E) 9 (E) 3 (E) – Ausdruck 29;

(B) – Ausdruck 10, 25, 20, 30, 25, 110.

ta 1978-02-08

Dipl Ing Gertried Tatzl, WIV
Mörikestraße 17, A-8052 Graz-Wetzelsdorf

Bedienungsanleitung

TABELLENAUFRECHNUNG
Datum 1979-03-08

TABELLENAUFRECHNUNG

1 START SPALTEN INDEX AUS SUMZ STORNO WERT Z

HP 97

Schritt	Anweisung	Eingabe	Tasten				Ausgabe	Forts.
1	Programm einlesen						0.	2
2	Bei Bedarf Dezimalstellen einstellen	(i)	DSP	(i)			0.i	3
3	Programm starten		f	A			1.i	4
4	Spaltenzahl eingeben	SP	SP	f	B		SP.i	5
5	Werte zeilenweise eingeben	W	W	E				6
	wenn kein Wert vorhanden: entweder 0 über die Taste E							
	eingeben oder den nächsten Spaltenindex über die							
	Taste A anwählen.							
6	letzter Wert einer Zeile ?							
	ja = automatischer Ausdruck Zeilensumme, wenn Index = SP,							
	wenn Index kleiner		E				1.i	5
7	Ausdruck Spaltensummen und Gesamtsumme		B				1.i	2
	Wahlweise können folgende Funktionen angewählt							
	werden :							
8	Anwahl des Spaltenindex	I	I	A			I.i	
9	Stornierung der letzten Eingabe		C				I-1.i	
	Storno-Taste nur einmal drücken !							
	Anm.: Mit i ist in der Spalte "Ausgabe" die Anzahl							
	der Dezimalstellen bezeichnet, mit der Summenausdruck							
	erfolgt							

ta 1978-02-08

Dipl. Ing. Gerfried Tatzl, WIV
Mörikestraße 17, A-8052 Graz-Wetzelsdorf

Programmlistung

Blatt-Nr. 1
HP 97

Programm TABELLENAUFRECHNUNG

Nr

Dat. 79-03-08

Schritt	Tasten	Code	Kommentar
1	* LBL a		Start, Speicher löschen, Index auf 1 stellen,
2	RCL E		Rettung max. Spaltenzahl
3	CL REG		
4	P⇄S		
5	CL REG		
6	STO E		
7	SPC		
8	* LBL 2		
9	1		
10	STO I		
1	CF 3		
2	RTN		
3	* LBL b		Eingabe max. Spaltenzahl (kleiner 23!)
4	STO E		
5	RTN		
6	* LBL A		Anwahl Spaltenindex (kleiner gleich SP!)
7	STO I		
8	RTN		
9	* LBL C		Stornierung der letzten Eingabe
20	1		
1	x = y?		
2	GTO 5		Fall Storno nach Ausdruck Zeilensumme
3	DSZ I		
4	RCL D		
5	CHS		letzter Wert wird negativ genommen
6	GSB 0		Durchführung der Stornierung
7	DZS I		
8	RCL I		
9	RTN		
30	* LBL 5		
1	R↑		
2	STO 0		letzte Zeilensumme zurückstellen
3	CHS		
4	PRT X		dieselbe negativ ausdrucken
5	RCL E		
6	STO I		letztmöglichen Spaltenindex einstellen
7	RCL D		
8	STO-0		Stornierung in der Zeilensumme
9	STO-(i)		Stornierung im Spaltenspeicher
40	RCL I		Anzeige letztmöglicher Spaltenindex
1	RTN		
2	* LBL B		Ausgabe Summenzeile Spaltensummen
3	SPC		
4	1		
5	STO I		Einstellung auf erste Spalte
6	0		wegen erster neuer Summenbildung nötig
7	* LBL 1		
8	RCL (i)		
9	PRT X		Ausdruck Spaltensumme
50	+		

Label: A Index; B Aus Sumz; C Storno; D; E Wert; a Start; b max. Sp; c; d; e

Dsp: FIX ☐ SCI ☐ ENG ☐ n: 0 x; 1 x; 2 x; 3 x; 4 x; 5 x; 6; 7; 8; 9

Speicher: P0 Su. Zl.; P1 Spalte; P2 "; P3 "; P4 "; P5 "; P6 "; P7 "; P8 "; P9 "; S0(10) "; S1(11) "; S2(12) "; S3(13) "; S4(14) "; S5(15) "; S6(16) "; S7(17) "; S8(18) "; S9(19) "; A(20) "; B(21) "; C(22) "; D(23) Wert; E(24) SP; I(25) Index

Trig: DEG ☐ GRAD ☐ RAD ☐

Flag: 0 ☐; 1 ☐; 2 ☐; 3 ☒ Eingabe

ta 1978-02-08

Dipl. Ing. Gerfried Tatzl, WIV
Mörikestraße 17, A-8052 Graz-Wetzelsdorf

Programmlistung

Blatt-Nr. 2
HP 97

Programm TABELLENAUFRECHNUNG

Nr.

Dat. 79-03-08

Schritt	Tasten	Code	Kommentar
51	ISZ I		
2	RCL E		
3	RCL I		
4	x y?		
5	GTO 3		zum Ausdruck Gesamtsumme
6	R		
7	R		
8	GTO 1		zurück zum Ausdruck nächster Spaltensumme
9	* LBL 3		
60	R		
1	R		
2	SPC		
3	PRT X		Ausdruck Gesamtsumme
4	GTO a		automatischer Start für neue Rechnung
5	* LBL E		zeilenweise Eingabe der Tabellenwerte
6	F 3 ?		Eingabeabfrage
7	GTO 0		bei Zahleneingabe zur Zeilensummenbildung
8	* LBL 4		keine Eingabe - Ausdruck Zeilensumme
9	RCL 0		
70	PRT X		Ausdruck Zeilensumme
1	0		
2	STO 0		Summenspeicher für neue Rechnung löschen
3	GTO 2		zum Beginn Eingabe neue Zeile
4	* LBL 0		Absummierung
5	STO 0		Sicherstellung für evtl. Stornierung
6	STO+0		Bildung Zeilensumme
7	STO+(i)		Bildung Spaltensumme
8	ISZ I		Einstellung auf nächste Spalte
9	RCL E		
80	RCL I		
1	x y?		
2	GTO 4		zum Ausdruck Zeilensumme wenn I = SP
3	CF 3		Löschen Eingabeflag
4	RTN		Stop und Anzeige nächster Spaltenindex
5			
6			
7			
8			
9			
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
0			

Label: A, B, C, D, E, a, b, c

Osp.: FIX ☐ SCI ☐ ENG ☐ n =: d, e, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Speicher: P0, P1, P2, P3, P4, P5, P6, P7, P8, P9, S0(10), S1(11), S2(12), S3(13), S4(14), S5(15), S6(16), S7(17), S8(18), S9(19), A(20), B(21), C(22), D(23), E(24), I(25)

Trig: DEG ☐ GRAD ☐ RAD ☐

Flag: 0 ☐, 1 ☐, 2 ☐, 3 ☐

ta 1978-02-08

Beispiel K: **Güteprüfung von Bordsteinen**

START	P ?			e, W
l_s	b,b_1,h,h_1,	1,G,P	V,RG,F,σ	$\sigma_{BZ} \rightarrow$ P

Als letztes Beispiel wollen wir die Lösung eines technischen Problems mit Hilfe von PROTAR erläutern.

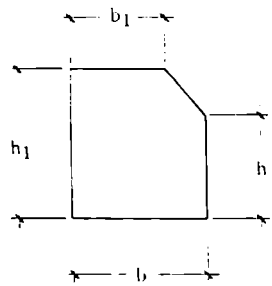
Die Darstellung der Funktionstasten sei wiederum an den Beginn gestellt.

START: Im allgemeinen ist für jedes Programm eine Startroutine zur Speicher-Freimachung erforderlich.

P ?: Eine bisher noch nicht erwähnte Variante, mit der die Ausgabe der Ergebnisse entweder über den Drucker, oder die Anzeige erfolgen kann. Hier steuert wiederum ein Flag diesen Vorgang. Die Anzeige selbst könnte über eine Pause laufen. Dies ist aber vor allem dann nicht zu empfehlen, wenn die Ergebnisse auch irgendwo einzutragen sind. In diesem Fall müßten Programmstops vorgesehen sein und zur Anzeige des nächsten Ergebnisses muß das Programm mit der R/S-Taste erneut gestartet werden.

l_s: Es ist der Abstand der Auflagerpunkte bei der Biegeprüfung einzutasten.

b,b_1,h,h_1: Es sind die tatsächlich festgestellten Querschnittsabmessungen



einzugeben.

l,G,P: Eingabe der Steinlänge, des Gewichtes und der gemessenen Bruchlast.

V,RG,F,σ: Durch Betätigung dieser Taste erfolgt die Berechnung und Ausgabe des Volumens des Prüflings, seines Festbetonraumgewichtes, des Faktors, mit der die Bruchlast multipliziert wurde, um zur Biegezugfestigkeit zu gelangen und schließlich die Biegezugfestigkeit selbst.

$\sigma_{BZ} \rightarrow$ P: Nach Eingabe der Abmessungen kann auch von einer eingegebenen Biegezugfestigkeit auf die dazugehörige Bruchlast rückgeschlossen werden.

Ursprünglich allein zur Auswertung von Güteprüfungen an Bordsteinen konzipiert, hat sich gezeigt, daß das Programm auch für andere Verwendungen herangezogen werden kann, wie dies im folgenden Absatz dargestellt wird.

Nach Eingabe der Querschnittsabmessungen kann nämlich die Lage der Schwerachse und des Widerstandsmomentes bestimmt werden, unabhängig davon, ob bei dieser Gelegenheit eine Güteprüfung durchgeführt wurde oder nicht. Werden diese Abmessungen um 90° gewendet eingegeben, d. h. für b und b_1 die Werte für h und h_1 und umgekehrt, lassen sich die gleichen Werte für die andere Lage bestimmen. Daraus ergibt sich automatisch auch die Lage des Schwerpunktes im Bordsteinquerschnitt. Der Ablauf des Rechenvorganges selbst ist durch die Diskussion der einzelnen Funktionstasten völlig klargestellt.

Es hängt sicherlich in erster Linie von der Gestaltung der Programme ab, inwieweit eine Problemlösung auch zur Lösung anderer, damit zusammenhängender Probleme herangezogen werden kann; oft sind nur einige Anweisungen mehr anzusetzen, um diesen Zusatzforderungen gerecht werden zu können.

Sachwortverzeichnis

Abarbeiten eines Programmes **31,** 93, 97
Abfragen **32,** 68
—, negative 68
Ablauf, normal 54
—, verzweigt 59
Ablaufplan, Ablaufschema 51, 53, **86,** 88, 101
Abruftaste 30
Adresse 23, **29,** 57
—, Label- 57, **58,** 82, 89
Adresstaste 29
algebraische Logik 17
Alpha-Label 58, **59,** 82
Analyse, Problem- 52, **83**
Änderungsdienst 102
Anweisungen **89,** 93
Anwendungsprogramm 116
Anzeige 15, **35**
Anzeigekapazität 34
Arbeitsgeschwindigkeit 49
Arbeitsspeicher 19, 20, **43**
Ausgabe **34,** 53, 112
Ausgabeformat 34
Auslistung, Programm- 97, 99, **100**
Auswahlkriterien 118
automatischer Zähler 44

Bedienerführung 35
Bedienungsanleitung 56, 58, 61, **99,** 100, 111
bedingter Sprung 32, 60, **63,** 69
Bedingung **32,** 60, 64, 68, 70, 72
Beendigung der Zahleneingabe 28
Befehl 89
Befehlsfolge 15
Beispiele 94
Betriebssystem 17
Bewegungsdaten 103
Buchstabenfolgen 23

Code, Tasten- **36,** 94
Codierblätter **94,** 101
Codierung 89

Darstellung, gepackte **42,** 103
Datenorganisation 106
Datenprogramm **107,** 126
Datenprogrammgenerator 107
Datensicherung **16,** 47
Datenspeicher 36, **37**
Datenspeicherbereich, Erweiterung des 38
Denkvermögen, logisches 10
Diagnoseprogramm 48
Display 35
Dokumentation 99
Druckeinrichtung 15
Drucker 35
Druckstreifen kleben 36

Eigenprogrammierung 116
Eingabe 112
—, Programm- 53, 55, 93
— während einer Pause 108
Einlesen 47, 96, **100**
Einsatzmöglichkeiten 115
Elemente der Programmierung 51
Energieversorgung 15
Erstfunktion 24
Erweiterung des Datenspeicherbereiches 38
ewige Schleife 76, 108, 109
Exponentialdarstellung **28,** 34

Fehlerausgang 81
Fehlerdiagnose 97
Fehlersuche 97
fixe Wortlänge 39
Fixkomma **28,** 34
Flag **46,** 68, 109
— Logik 109
flow chart 88
Formblattwesen 112
freie Funktionstaste 82
Funktionstasten, mathematische 26
—, Mehrfachverwendung von freien 107

gepackte Darstellung **42,** 103
Gleitkomma **28,** 34

Hauptprogramm **76,** 78, 111

Indexregister 45
indiziertes Rechnen 44
Individualprogramm 116

Kapazität 89
—, Anzeige- 34
—, Programmspeicher 60

Kettenrechnung 28
Kleben Druckstreifen 36
Kommentar 101
Konnektor 54
Konstantspeicher 46
Korrektur von Programmen 32

Label 48, **57**, 77, 87, 89
— Adresse **57**, 58, 89
—, Alpha- 57, **58**, 82, 89
—, numerischer **58**, 82
—, Primär- **59**, 82
—, Sekundär- **59**, **82**
Labeltaste 57
Laden 96, 100
LAST X-Register 47
Leerbefehl 63
Listungen **47**, 100
—, Programm- 97, 99, **100**
Logik, algebraische 17
— der umgekehrten polnischen Notation 19
logisches Denkvermögen 10
Löschen 30
Löschtasten 30

Magnetstreifen **47**, 96, 100
Marke 48, 53, **57**
mathematische Funktionstasten 26
Mehrfachverwendung von Datenspeichern 44, **103**
— von freien Funktionstasten 107
Modul 48
Modus 55

Negativabfrage 68
Null-Abfragen 32
numerischer Label **58**, 82

Operation 53
Operationsprogramm **107**, 126
Optimierung eines Programmes 98
Organisation einer Programmierung 115

Packen **42**, 43
Papiervorschub 34
Pause **35**, 71, 108
Präfixtaste **24**, 33
Primärfunktion 24
Primär-Label **59**, 82
Problemanalyse 52, **83**
Programm **51**, 81
Programmabschluß 56
Programmanfang 56
Programmbefehlstasten 31
Programmbeschreibung 99, **101**
Programmbibliothek 116
Programmdokumentation 99
Programmeingabe 53, 55, **93**
Programmiersprache 51
Programmierung, Elemente der 51
Programmschleife **69**, 75
Programmschritte **36**, 89
Programmspeicher 36
Programmspeicherkapazität 60
Programmspeicherung 96
Programmtest 94
Programmzeiger 111
Programmzweig 61, 71, **82**, 111
Programmweiche 60
Programm-Listung 36, 97, 99, **100**
– Marke 53, **57**
– Modus **55**, 93
Prozentrechnung 29

Rechengeschwindigkeit 49
Rechenlogik 15, **17**
Rechenzeit 110
Rechen-Modus **5**, 94
Rechnen, indiziertes 44
Rechner **14**
Rechnermodelle 152
Redigierung 98
Register 36
—, LAST X- 47
Routine 96
Rücksprungadresse 77
Run-Modus **55**, 94

Schalter 65
Schleifen **69**, 71, 75
Schleife, ewige 76, 108, 109
Sekundärfunktion **24**, 33
Sekundär-Label **59**, 82
Sequenz 15
Speicher 15, **36**
—, Mehrfachverwendung von 44
—, Mehrfachbelegung von 103
—, sonstige 43
Speicheradresse **29**, 38, 57
Speicherarithmetik 37
Speicherbefehle 29
Speichertauschtasten 30
Speicherteilung 39
Speicher-Test 87

Speicherung, Programm- 96
Sprache 51
Sprung 69
—, bedingter 32, 60, **63,** 69
—, unbedingter **61,** 69
— ins Unterprogramm 77
Sprungadresse **61,** 77
Sprungbefehl 45, **61**
Stackdrop 21
Stackspeicher 19, **43**
Standardprogramme 116
Start-Label 58
Steuerprogramm 107
Stromausfall 15
Symbole 52, **53**

Taste, Label- 57
—, Präfix- 33
—, sonstige 33
Tasten-Code **36,** 94
Tastenfeld 15, **23,** 24, 25
Tastenprogrammierung 52
technische Zahlendarstellung 28
Test 94
Testbeispiel 101
Test-Register **44,** 87
Texte 23
Thermodrucker 35

Überarbeitung 98
umgekehrte polnische Notation 19
unbedingter Sprung 69
Unterprogramm 43, 49, **76,** 78, 90, 111
Unterprogrammebenen 80
Unterprogramme, Wirtschaftlichkeit von **78,** 79

Vergleichstasten 32
Verzweigungen 46, 53, **59,** 63, 86, 97
Vorwahltaste **24,** 33

Wahrheitsgehalt 97
Weichen, Programm- 60
wissenschaftliche Zahlendarstellung 28
Wirtschaftlichkeit von Unterprogrammen **78,** 79
Wortlänge 39

Zahlendarstellung, gepackte 103
Zahleneingabe, Beendigung der 28
Zahlenfeld 26
Zahlenformat 28
Zähler, automatischer 44
Zeilennummer **36,** 57, 58, **93**
Zerlegen gepackter Zahlen 43
Zweitfunktion 24

Der Taschenrechner als Mini-Computer

Band 2: Kalkulation in Produktionsbetrieben mit programmierbaren Elektronenrechnern

Von Dipl.-Ing. G. Tatzl

Ca. 100 Seiten DIN B 5. Kartoniert ca. DM 19,—

Mit diesem Buch werden neutrale Softwareinformationen für Mini-Computer weitergegeben. Dem in der Nutzung programmierbarer Kleinrechner geübten Anwender wird es ein leichtes sein, sein individuelles Kalkulationsproblem aufgrund der gegebenen Anregung zu lösen. Dem weniger geübten Leser wird dieses Buch aber eine Anregung sein, sich näher mit der Technik des Programmierens zu befassen, weil er während der Lektüre leicht erkennen wird, wozu Mini-Computer fähig sind und welche Vorteile er selbst ziehen kann, wenn er die Durchführung von Kalkulationen im Hinblick auf die Rechentechnik einem programmierbaren Kleinrechner überträgt.

Die Wahl des richtigen Taschenrechners

Von Verm.-Ing. H. Osterloh

64 Seiten DIN B 6. Kartoniert DM 8,—

Ebenso, wie heute jeder Jugendliche und Erwachsene eine Uhr besitzt, so wird bald auch der Besitz eines Taschenrechners ähnlich verbreitet sein. In vielen Grundschulen gelten solche Geräte schon jetzt als Lehrmittel, aus der Berufspraxis sind sie nicht mehr wegzudenken.

Nun stellt sich aber mehr denn je für den Käufer das Problem, aus der Fülle von Gattungen und Preislagen der angebotenen Taschenrechner das für den eigenen speziellen Bedarf am besten geeignete Gerät auszuwählen.

Horst Osterloh, ein erfahrener Ingenieur und Verfasser von mehreren in vielen großen Auflagen verbreiteten Tabellen-Büchern, hat es deshalb übernommen, die wichtigsten Aspekte für den Kauf von Taschenrechnern, angefangen von den vier Grundrechenarten bis hin zu den programmgesteuerten Tischrechnern, übersichtlich und konzentriert zusammenzutragen, ohne sich dabei in Einzelheiten zu verlieren.

Hauptaufgabe dieses neuen Büchleins ist es, eine allgemeine Übersicht und bei der Auswahl eines Taschenrechners die notwendige Entscheidungshilfe zu geben. Dazu werden auch einige typische Rechenbeispiele angeführt.

Die Hinweise sind absolut firmenneutral gehalten, d.h., es wird nicht auf einzelne Fabrikate eingegangen. Vermittelt werden vielmehr die notwendigen Kenntnisse für die bedarfsgerechte Auswahl von Taschenrechnern, um vor Fehlinvestitionen bzw. Enttäuschungen zu bewahren.

Angesprochen werden alle Bedarfsgruppen, vom Oberschüler über den Techniker und Kaufmann bis hin zum Wissenschaftler. Das Büchlein enthält auch wichtige Hinweise für die richtige Ausrüstung von Büros und Unternehmen mit Taschenrechnern.

BAUVERLAG · WIESBADEN UND BERLIN

Projekt-Management

Planung und Abwicklung von Projekten

Von Prof. Dr.-Ing. W. Dreger

XXIV, 322 Seiten DIN A 5 mit zahlreichen Abbildungen und Tabellen. Gebunden DM 75,—

Organisatorische und kreative Fähigkeiten bilden die Voraussetzungen für erfolgreiches Projekt-Management (PMT) in der Industrie und in den Projektierungsbüros. Dem Projekt-Manager obliegt es hierbei vor allem, die Zielsetzungen für die Projekte zu formulieren. Dafür hat er interdisziplinäre Projektgruppen zu bilden, in denen Angehörige aus den verschiedensten Funktionsbereichen und Entscheidungsebenen gleichberechtigt an der Lösung gestellter Aufgaben (Projekte) in den Bereichen Produktion und Verkauf zusammenarbeiten.

Qualifikation, Mobilität und Kreativität im Team haben im Projekt-Management höchste Bedeutung. Sie sind schließlich eine weitgehende Garantie für das bestmögliche, mit herkömmlichen Verfahren kaum so effektiv erreichbare Endergebnis.

Eine ausführliche Anleitung für erfolgreiches Projekt-Management bietet dieses neue Buch. Es zeigt Möglichkeiten der Projekt-Organisation und der Projektablaufplanung auf und vereinigt in sich die Erfahrungen aus bereits realisierten Projekten. Darüber hinaus enthält es schriftliche Formulierungen, die bei Unklarheiten und Streitigkeiten innerhalb des Projekt-Managements die klare Handhabung aller Detailaufgaben ermöglichen.

Damit vermittelt das Handbuch die Voraussetzungen für den optimalen Einsatz – mit und ohne Netzplantechnik – von Arbeitskräften und Material bei mittleren und größeren Projekten aller Art, die mit Hilfe des Projekt-Managements sinnvoll und rationell geplant bzw. abgewickelt werden können.

Leasing-Kompendium der Praxis

Von Dipl.-Volkswirt Dr. G. Feist.

164 Seiten DIN A5 mit zahlreichen Tabellen. Kartoniert DM 40,—

Der Autor, freiberuflicher Berater mittlerer und kleinerer Unternehmungen, durch Fragen zum Thema aus dem Teilnehmerkreis vieler Seminare geschult, bietet in allgemeinverständlicher Form Hilfen für vergleichende Berechnungen darüber, ob ein Ausrüstungsgegenstand oder eine Maschine durch Leasing oder auf herkömmliche Art finanziert werden sollte.

Vor allem wird hier das „Mobilien-Leasing" (leasen von Maschinen und Gerät) behandelt, aber auch das Immobilien- und Personal-Leasing werden erwähnt. Alle Finanzierungsberechnungen werden durch grafische Darstellungen und Hinweise auf die steuerliche Belastung unterstützt und leichter faßbar gemacht. Ein Buch, das jeder Verantwortliche **vor** Anschaffung eines kostspieligen Wirtschaftsgutes machen sollte.

BAUVERLAG · WIESBADEN UND BERLIN

Grundstücks- und Gebäudebewertung marktgerecht

mit Formeln, Rechenverfahren, Diagrammen, Tabellen und Rechner-Programmierung

Von Ing. (grad.) M. Vogels, Architekt

240 Seiten DIN B 5 mit 38 Abbildungen und 69 Tabellen. Gebunden 58,—

Aufgabe dieses Buches ist die praxisgerechte Darstellung von zeitgemäßen Verfahren zur Ermittlung marktgerechter Grundstückswerte. Es werden sowohl einfache Methoden zur überschlägigen Wertermittlung oder Kontrolle als auch die üblicherweise von Fachleuten benutzten anspruchsvolleren Verfahren ausführlich behandelt.
Die hier im Gegensatz zu vielen althergebrachten Verfahren in wesentlicher Verfeinerung gebotenen Berechnungsmethoden und zahlreiche Beispiele machen auch dem mathematisch weniger versierten Leser selbst komplizierte Zusammenhänge schnell deutlich. Verschiedene Diagramme und umfangreiche Tabellen begrenzen den notwendigen Rechenaufwand auf ein Minimum.
Behandelt werden alle Bereiche der Wertermittlung von unbebauten und bebauten Grundstücken, wie z. B. Bodenwert, Bauwert, Ertragswert, Verkehrswert, Grundstücksbelastung, Unternehmensbewertung, Beleihungswert. Der Autor geht aber auch auf die Honorare und auf spezielle Sonderfälle, wie z. B. Erbbaurechte, Leibrenten, Sozialer Wohnungsbau, Nutzungsbeschränkung, Nießbrauch usw. ein. Er zeigt ferner die heute so wichtige und oftmals viel zu wenig bekannte Anwendung statistischer Methoden bei der Wertermittlung und die Nutzung moderner elektronischer Rechengeräte.

Verrentung von Kaufpreisen Kapitalisierung von Renten

Tabellen mit Berechnungsgrundlagen von Leibrenten, Zeitrenten und dynamischen Renten

Von Ing. (grad.) M. Vogels, Architekt

Ca. 75 Seiten DIN B 5 mit zahlreichen Tabellen. Gebunden ca. DM 20,—

Ziel dieser Broschüre ist es, Tabellenmaterial zur Verfügung zu stellen, das eine einfache, unkomplizierte und damit schnelle Kapitalisierung von Renten oder Verrentung von Kaufpreisen ermöglicht.
Die Leibrenten-Tabellen wurden deshalb für die häufigsten Fälle errechnet, nämlich bei Einzelpersonen für monatlich vorschüssige Zahlung und bei Ehepaaren für monatlich vorschüssige Zahlung bis zum Tod des Letztlebenden. Der Zinsbereich von 3 bis 7% sowie die Altersunterschiede bei Ehepaaren zwischen +10 und —10 Jahren dürften den größten Teil der zu lösenden Aufgaben abdecken. Anwendung von Formeln oder Interpolation ist hier nicht mehr nötig.
Für weniger häufig vorkommende Fälle werden Beispiele und Rechenwege (Formeln) angegeben, die auf den abgedruckten Tabellen aufbauen und nach einiger Übung die Lösung der Probleme ermöglichen.

BAUVERLAG · WIESBADEN UND BERLIN

Kostenplanung für Hochbauten

Praxisbezogene Planungs- und Steuerungsinstrumente für mehr Kostensicherheit und Wirtschaftlichkeit

Von W. Jendges, Architekt

190 Seiten DIN A 5 mit zahlreichen Tabellen. Gebunden DM 52,—

Bau- und Betriebskosten von Büro- und Verwaltungsbauten

Eine Auswertung der Daten von 110 ausgeführten und in Betrieb genommenen Gebäuden

Von Prof. Dr. Ing. C. Siegel und Dipl.-Ing. R. Wonneberg

2., durchgesehene Auflage. 365 Seiten DIN B 5 mit 53 Tafeln und 110 Seiten Zeichnungen im Dokumentationsteil. Kartoniert DM 70,—

Kalkulation von Baupreisen

Von Prof. Dr. Ing. G. Drees, Dipl.-Ing. P. Haller und Dipl.-Ing. B. Kochendörfer

305 Seiten DIN B 5 mit 79 Abbildungen. Gebunden DM 75,—

KLR Bau

Kosten- und Leistungsrechnung der Bauunternehmen

Herausgegeben vom Hauptverband der Deutschen Bauindustrie e.V. und dem Zentralverband des Deutschen Baugewerbes e.V.
2., durchgesehene und erweiterte Auflage. 128 Seiten DIN A 4 mit zahlreichen Tabellen und acht Faltblättern. Gebunden DM 78,—

Die „KLR Bau" bildet zusammen mit dem Baukontenrahmen ein geschlossenes System des baubetrieblichen Rechnungswesens. Dabei stellt die Kosten- und Leistungsrechnung das Herzstück des baubetrieblichen Rechnungswesens dar. Dies gilt insbesondere für die Bauauftragsrechnung als teil der KLR. In der Bauauftragsrechnung wird vorrangig die zum Angebotspreis führende Vorkalkulation unter modernen baubetriebswirtschaftlichen Gesichtspunkten dargestellt. Der Bau-Angebotspreis von heute entscheidet im Auftragsfalle über den Erfolg der Baustelle von morgen. Das Baustellenergebnis wird über die Baubetriebsrechnung zahlenmäßig sichtbar und nicht nur für Zwecke der Nachkalkulation, sondern auch für die Vorkalkulation künftiger vergleichbarer Bauobjekte bedeutsam. In der KLR Bau wird eine Brücke zwischen den technischen und kaufmännischen Abteilungen des Bauunternehmens geschlagen und für beide eine gemeinsame Sprache in den Begriffen und ihren Inhalten geschaffen.

BAUVERLAG · WIESBADEN UND BERLIN

Freihändig zeichnen und skizzieren

Lehr- und Arbeitsbuch

Von Prof. Dipl.-Ing. K. Ch. Heuser, Architekt

3. Auflage. 163 Seiten DIN A 4 mit zahlreichen Zeichnungen und Übungsskizzen. Kartoniert DM 28,—

Dieses Lehr- und Arbeitsbuch dient mit seinen ungeahnt vielseitigen und aufschlußreichen Beispielen sowie mit seinen fachlich klaren Hinweisen und Übungsblättern dazu, die handwerklichen Grundkenntnisse und somit auch die angestrebte Sicherheit in der technischen Ausführung von freihändigen Zeichnungen zu vermitteln. Es ist sowohl für Hobbyzeichner als auch für die Berufsausbildung und -praxis bestimmt, zumal es heute kaum noch einen Berufszweig gibt, in dem nicht die Beherrschung des freihändigen Zeichnens und Skizzierens von großem Nutzen ist.

Energiesparen jetzt!

Arbeitsmethoden und Checklisten zum Kostensenken in bestehenden und neuen Gebäuden und Industrieanlagen

Von M. G. Kiss, H. P. Mahon und H. J. Leimer

312 Seiten DIN B 5 mit zahlreichen Abbildungen und Tabellen. Gebunden DM 98,—

Dieses Buch bietet eine Zusammenfassung der Energiespartechnik nach neuestem Stand. Die Autoren verfügen über langjährige und umfassende Erfahrung aus Forschung, Planung und Ausführung bei der Energieeinsparung. Miklós G. Kiss ist als leitender Ingenieur für die Planung von Gebäude- und Industrieinstallation, Harold P. Mahon als Professor für angewandte Physik an der University of Massachusetts, Boston, und Hansjörg Leimer als leitender Ingenieur für die Planung von Energieversorgungsanlagen und für Alternativenergien tätig.

Im ersten Teil des Buches werden die wesentlichsten Aspekte der Energieeinsparung, Energiekonzepte und die Organisation von Energiesparprogrammen beschrieben. Hierzu bieten die Autoren praxisgerechte Checklisten für mehrstufige Energiesparmaßnahmen, die bereits in ihrer ersten Stufe eine Energieeinsparung von 10–20% erbringen können.

Für Architekten, Ingenieure, Betriebsleiter und für die Energiebeauftragten in den Unternehmen ist der zweite Teil des Buches besonders von Bedeutung. Hier werden bereits durchgeführte Energiesparprogramme mit Angaben über die Höhe der Kosten und der erzielten Einsparungen beschrieben. Ergänzend werden Grundsätze und Berechnungsbeispiele für die energiesparende Planung von neuen Gebäuden gegeben.

BAUVERLAG · WIESBADEN UND BERLIN

Gertried Tatzl
Dipl.-Ing. WIV,
Jahrgang 1932

Studium des Wirtschaftsingenieurwesens, Fachrichtung Bau, an der Technischen Universität Graz.

Leitende Tätigkeit in Kostenrechnung und DV-Organisation in der Industrie. Initiativ am Aufbau eines modernen Kostenrechnungs- und Kalkulationssystems beteiligt. Mitarbeit an Bearbeitung von Fragen der Technologie und Güteprüfung von Baustoffen. Seit 1975 intensiv mit Fragen der Anwendung von kleinen Informationssystemen befaßt.

Zu diesem Buch

Die Entwicklung der Rechentechnik hat in der letzten Zeit nicht nur bei Großrechenanlagen, sondern auch bei kleinen Systemen, wie den Taschen- und Tischrechnern, einen bisher nicht gekannten Leistungsstandard gebracht. In der Praxis zeigen sich nicht nur für technische, sondern auch für kommerzielle Bereiche noch nicht ausgeschöpfte Anwendungsmöglichkeiten für Mini-Computer.

Dieser Entwicklung Rechnung tragend, wird im vorliegenden Buch und in den weiteren Bänden des Gesamtwerkes in leicht verständlicher Weise einem Personenkreis, der in der DV nicht bewandert ist, Hilfestellung in optimaler Nutzung und Programmierung von Mini-Computern geboten. Vor allem gilt dies für jene, die sich bisher mit konventionell arbeitenden Rechnern plagen mußten und nun ihre Probleme wesentlich rascher lösen können.

ISBN 3-7625-1213-3